PRivacy Enabled Capability In Co-Operative Systems and Safety Applications

Deliverable 7

V2X Privacy Verifiable Architecture

Project:	**PRECIOSA**
Project Number:	**IST-224201**
Deliverable:	**D7**
Title:	**V2X Privacy Verifiable Architecture**
Version:	**v2.0**
Confidentiality:	**Public**
Editor:	**Frank Kargl**
Cont. Authors:	**Stefan Dietzel, Lukas Dölle, Johann Christoph Freytag, Martin Kost, Antonio Kung, Zhendong Ma, Florian Schaub**
Date:	**02.11.2009**

Part of the Seventh Framework Program
Funded by the EC-DG INFSO

Document History

Version	Status	Date
v0.1	Initial version	15.12.2008
v0.2	Contribution on design process	20.01.2009
v0.3	Integrating contributions from various partners	01.04.2009
v0.4	First completed draft version	05.04.2009
v0.5	Reviewed and revised version	07.04.2009
v1.0	Prepared version for submission	10.04.2009
v1.1	Started major restructuring and revision	25.05.2009
v1.2	Revised version for project internal discussion	13.06.2009
v1.3	Integrating revised architecture	12.08.2009
v1.9	Draft version for project internal review	05.10.2009
v2.0	Final version incorporation reviewer feedback	25.11.2009

Approval		
	Name	Date
Prepared	Frank Kargl	25.11.2009
Reviewed	All Project Partners	26.11.2009
Authorized	Antonio Kung	27.11.2009

Circulation	
Recipient	Date of submission
Project Partners	01.12.2009
European Commission	01.12.2009

Contents

List of Figures

List of Tables

1 Introduction

1.1 Motivation

It is the idea of PRECIOSA to provide a so called “privacy-verifiable architecture”, i.e., an architecture that can guarantee certain privacy properties and these properties can be verified by some external entity, e.g., a user or trusted third party.

This contrasts with today’s situation where most systems in use basically allow the data processor to declare its privacy policy and intended use to the data subjects, i.e., the persons providing the data. P3P [3] is an example of such an approach, where a website can declare its privacy policy and how data submitted by a web user will be utilized. The user (or a specifically configured browser) can then check whether the policy is compatible with the own privacy requirements and decide whether to trust the data processor and submit the personal information or to not reveal personal information and end the session. This puts the user in a very weak position. He can either accept the policy of the data processor and submit data, or reject the policy and give up using the offered service.

Moreover, various incidents in the past[1] have shown that this declarative approach is not sufficient. Most of the time, there is no control whether the data processor actually adheres to its stated policy. But even if the data processor intends to comply to its policy, if data is not sufficiently protected, data can get lost or an attacker can steal the data[2] resulting in uncontrollable use of the data.

For future cooperative ITS (cITS) systems, this is not an acceptable solution for various reasons:

1. Data processed by ITS systems is very privacy sensitive, as it reveals locations and movement patterns of drivers (see PRECIOSA deliverables D1 and D4). Loss of this data or submission for uncontrolled use is not acceptable.
2. Drivers are not expected to know details about the applications running in the ITS and are not capable of verifying privacy policies when using a new service because they are not familiar with privacy details. Even more importantly, drivers must not be distracted by configuration issues during driving.

[1] E.g., the Deutsche Telekom collected and scrutinized call data of journalists and members of the supervisory board during 2005 and 2006 – http://www.time.com/time/business/article/0,8599,1809679,00.html; in 2009 the chief of the German railway (Deutsche Bahn) finally offered his resignation after active manipulation of trade union leaders’ e-mail traffic became public – http://www.timesonline.co.uk/tol/news/world/europe/article6004352.ece

[2] E.g., the British tax agency lost personal and financial data of 25 million citizens in November 2007 – http://news.bbc.co.uk/2/hi/uk_news/politics/7104945.stm

3. There is no guarantee whatsoever that the data processor will actually adhere to the privacy policy and process data only in the specified way. The data subject has to ultimately trust the data processor.

Therefore, we take a completely contrary approach for the PRECIOSA V2X Privacy-Verifiable Architecture. We start from the privacy requirements of the data subject, i.e., the user. The user uses a specific policy language or other tools that generate this language to express the purpose he provides his data for. Our architecture then provides a cryptographically secured path from the data subject to a specific application and this application must contain an access control mechanism that only allows data access in a policy compliant way. Access control must be mandatory and it must not be possible to circumvent it.

The resulting privacy-aware and privacy-protected cITS is also termed *hippocratic cooperative ITS* or *HcITS* with reference to hippocratic databases as defined by Agrawal e.a. [4]. Our starting point for defining an HcITS are the privacy requirements of the data subject expressed as a privacy policy. The HcITS architecture is then designed in a way that prevents any non-compliant data access.

The HcITS architecture presented here consists of three components:

1. *Architecture Principles* provide the basic ideas that our design is based upon. Architecture principles are directly derived from *Architecture Requirements*.
2. The *cITS Design Process* describes a methodology that should be followed when designing HcITS. This includes guidelines and methods ensuring that a “privacy by design” approach is taken when planing for new cITS applications.
3. A *cITS Runtime Architecture* that provides mechanisms that will ensure protection of personal information and compliance with privacy policies during runtime.

Both the design process and the runtime architecture will be presented in two steps:

1. The *Cooperative ITS Architecture Framework* (cITS AF) is a generic and reusable framework that substantiates the architecture principles. Whereas the AF provides the basic architecture of our later prototype, it only describes the basic building blocks and their functionality without detailing how this functionality is actually implemented. So the AF is really a framework for building privacy-verifiable architectures in the ITS domain but its basic structure can easily be transfered to other domains like Internet and Web or other forms of Ubiquitous Computing systems.
2. The *Cooperative ITS Reference Architecture* (cITS RA) provides a specific instantiation of the cITS AF that will later be used to implement a prototype privacy-verifiable ITS based on the use cases from deliverable D1 [5]. The reference architecture already specifies the mechanisms that will be used in the implementation to the extent needed on the architecture level. Complete specifications and analysis are later provided by D10 and other deliverables.

1.2 Document Structure

The document is structured as follows: Chapter 2 will outline requirements and principles that our architecture is built upon. Chapters 3 and 4 discuss the design process of privacy-verifiable cITS applications, where chapter 3 discusses the architecture framework and chapter 4 focuses on the reference architecture. Likewise, chapters 5 and 6 present the runtime architecture, with chapter 5 discussing the architecture framework and chapter 6 the reference architecture. Finally, Chapter 7 summarizes this deliverable with concluding remarks.

2 Requirements and Principles

This chapter discusses the requirements that our architecture must fulfill and what generic architecture principles can be derived from it.

2.1 Architecture Requirements

In this section we describe the basic requirements that our architecture must fulfill. These requirements will govern the structure and functionality of both the design process and runtime architecture that are described later in these documents and therefore need to be highlighted first.

We first describe some functional requirements that directly relate to the privacy domain that the PRECIOSA architecture should address. Next, we also discuss two additional aspects that have strong impact on architecture, as they are derived from the operational environment in which the PRECIOSA architecture is to be embedded.

2.1.1 Functional Requirements

In order to comply to the idea of a privacy-verifiable architecture, our architecture must fulfill the following requirements. These can also be seen as the strategic goals of PRECIOSA.

- The architecture should support both the *design-time* and *run-time* of cITS systems. At design-time, the architecture should support design of privacy-friendly ITS and comparison of design alternatives to chose more privacy-friendly solutions.
- The architecture should consider *organizational*, *jurisdictional*, as well as *technical* issues. This can be addressed by a design methodology that ensures such consideration.
- The architecture should address privacy in all domains of the ITS providing *system privacy* instead of only communication or storage privacy.
- Privacy should become *measurable*, i.e., comparison of different systems and design alternatives should be possible in a measurable and objective way that is not only based on subjective judgment.

- Privacy should become *enforceable*, so that data subjects do not need to rely only on the declared intent of data processors but gets some guarantees that the policies they set are technically enforced by the system.

These architectural requirements need to be combined with application and validation requirements presented in deliverable D8 “V2X application specification and validation requirements”. Deliverable D11 “Guidelines for privacy aware cooperative application” will provide detailed privacy guidelines for cITS applications beyond what is presented here. These guidelines must also be fulfilled either by this architecture or by system designers.

2.1.2 Additional Considerations

Embedded Systems Software Deployment

ITS applications are systems which include embedded systems (e.g., a telematics box in a vehicle, a road side unit). Many such systems have resource constraints and therefore their design is in many cases static, i.e., the resources are predetermined at design time.

While this approach lacks flexibility with respect to architectures where we can dynamically create resources, it brings two benefits: systems are more aware of resources and they are often more deterministic. For example, in the AUTOSAR engineering approach,[1] engineers configure at build time the resources needed by electronic control units in a vehicle. This could involve constants that can no longer be changed.

Furthermore, this means that there are a number of configuration parameters that are hardwired and unchanged. Privacy friendly ITS applications imply that such embedded systems must integrate features that will ensure that configuration parameters related to privacy just follow the same approach. Such configuration parameters include in particular privacy policies. A privacy policy could be the following: “erase this data after one hour”. In a static system this policy could be hardwired, meaning that once the system is built up, the policy can never be changed. In more dynamic systems, this policy could be data meaning that there is the capability to change the policy.

Business stakeholders’ separation of concerns

Future ITS applications will be deployed as businesses. It is therefore important to understand how those future businesses might impact the PRECIOSA architecture. Future ITS infrastructures will allow many independent ITS applications to run in parallel (e.g., tolling, insurance, traffic information, ...). They will likely share some of the computing resources (e.g., a shared telematics box, a shared RSU, ...). From an application viewpoint, the architecture must ensure that each component of the system is clearly assigned to a stakeholder. By associating components with stakeholders, a possible (external) attacker must interact with this stakeholder and the responsibility for privacy protection can

[1] AUTOSAR website: www.autosar.org

be assigned to this stakeholder. This links attackers of components, whether external or internal, clearly to the persons involved with this component. Privacy issues can then be linked to the respective responsible stakeholder and it is their task to provide a trust model for the application users.

While the above principle mostly holds for data storage issues, information exchange by data transmission must also be linkable to responsible organizations. In contrast to data storage, in the area of information exchange a major threat arises from eavesdroppers. Their access to information while it is transmitted is more difficult to limit than for stored data. During the exchange of information, confidentiality and identification and verification of the recipient is required to ensure that information is accessible only by the intended destination.

In the design phase of an ITS application, the architecture must reflect that an attack may arise in anytime when data are stored or transmitted. In all cases, countermeasures to attacks must be possible and explicitly be foreseen. Extensions to the “basic” functional architecture by privacy-enhancement features must be foreseen, i.e., the existing data flows must be protected by privacy enhancing functionalities so that unintended data flows are suppressed. This sharing also brings the issue of multi-level security. Different sets of possibly conflicting access requirements might have to be managed. For instance, an application would require the retention of a data item while a data subject’s policy requires its deletion.

2.2 Hippocratic cITS Principles

The purpose of this section is to provide guidance for a generic (and later detailed) architecture that helps us to implement a privacy aware architecture for ITS. We therefore introduce several principles that have been proposed in the literature and adapt them to the requirements of ITS. Based on those principles we develop an example showing that more aspects need attention. Those aspects relate to the metadata that must be stored and managed to reflect the privacy policies and preferences of the different participating actors as well as to requirements addressing how the software development cycle must be reviewed in the context of privacy aware systems.

2.2.1 Motivation and founding Guidelines

In the following we present general principles that should guide the implementation of any privacy aware ITS System. They are motivated by several publications, in particular by the following two documents:

1. In 2003, R. Agrawal, J. Kirnan, R. Srikant, and Y. Xu describe in their paper *Hippocratic Databases* a set of principles that should guide any privacy aware database management system (DBMS). We extend this paper for Hippocratic cooperative ITS systems (HcITS) that should provide the basis for our approach [4]. These principles

will be used as a basis in Subsection 2.2.2 for defining the hippocratic principles for cooperative ITS, and are, therefore, not explicitly state here.

2. In its Subsection 7.2 the ISO Technical Report TC 12859 *Intelligent transport systems - System architecture - Privacy aspects in ITS standards and systems* provides recommendations "*. . . under which data shall be collected and held in support or provision of ITS services . . .*" [6]. Those principles are
   - Avoidance of harm,
   - Fair and lawful processing of data,
   - Data collection for specified, explicit, and legitimate purposes,
   - Explicit purpose and legitimacy of collection must be determined at the time of collection of the data,
   - No further data processing in a way that is incompatible with the purposes for which it was originally collected,
   - No disclosure of data without the consent of the data subject,
   - Data collection must be adequate, relevant, and not excessive in relation to the data collection purposes,
   - Data is kept accurate and, where necessary, up to date,
   - Identification of data subject for no longer than necessary for the purposes for which the data was collected,
   - Restriction of data collection to those who have demonstrable "need to know",
   - Clear and accessible practices and policies must be maintained by the data collectors,
   - Data protection must be assured by security safeguards,
   - Recommendations are allowed only in a cumulative manner.

2.2.2 The principles of Hippocratic cooperative ITS

As PRECIOSA is driven by the aspects and abilities of current technology our principles are rooted heavily in the recommendation of the ISO Technical Report and in the paper by Agrawal et al. Our principles should clearly articulate in an implementation independent manner what it means for a cooperative ITS system to manage personal information under its control responsibly. Furthermore, the principles also express for users of a Hippocratic cooperative ITS (HcITS) what they can expect from the system without being technically adept.

We define the principles for HcITS as follows:

1. **Purpose specification:** for all personal information that is communicated between or stored by participating components of an ITS system the purposes for which the information has been collected shall be associated with that information.

   The purpose might be explicitly or implicitly specified by the donor – as the data subject is called here – or might be derivable from the current operational context. Furthermore, we expect the cITS to answer questions of the data subject such as why specific information is being operated on.

2. **Consent:** the donor of the information must provide his/her consent for the usage of information for a specified purpose. This consent might be restricted to one specific purpose or to a set of purposes; the data subject should also have the right and ability to revoke his consent for the future.

3. **Limited Collection:** Information related to the data subject (i.e., information describing properties or aspects of the data subject) shall be limited for communication, storage, and collection to the minimum necessary for accomplishing the specified purposes.

4. **Limited use:** The cITS shall execute only those operations on the personal information that are consistent with the purposes for which the information was collected, stored, and communicated.

5. **Limited Disclosure:** The personal information related to the data subject and operated on by the cITS shall not be communicated outside the cITS for purposes other than those for which the data subject gave his/her consent.

6. **Limited Retention:** Personal Information related to the data subject shall be retained by the cITS only as long as necessary.

7. **Accuracy and Context Preservation:** Personal Information related to the data subject and stored by the cITS shall always be accurate, up-to-date, and never be decoupled from its context and purpose.

8. **Security:** personal information related to the data subject shall be protected by appropriate security measures against unauthorized use or use in conflict with the consent of the data subject.

9. **Openness:** A data subject shall be able to access all information that is stored in the cITS and is related to the data subject.

10. **Compliance:** A data subject shall – directly or indirectly – be able to verify the compliance of the CITS with the above principles.

In the following we briefly discuss the principles and give examples for some of them.

The Principle of *Purpose Specification* is tightly bound to the information provided by the data subject, therefore this additional information – called describing data or metadata – should not be separated as many of the other principles rest on that metadata. Of course this purpose specification is closely linked to Principle 2 (Consent). However, the Principle of *Consent* raises the question of how to resolve possible conflicts between the

purpose specification and/or governmental or legal requirements that might contradict the data subject's consent.

The Principle of *Limited Collection* requires deciding carefully which data is really needed to perform what kind of service. For example, if a data subject driving a car performs a hotel reservation, it might be necessary to provide a credit card number. However, it does not seem to be necessary to provide the current location of the car when making a hotel reservation.

The Principle of *Limited Use* means to obey the purpose specification and the consent given by the data subject. This should be taken as one of the correctness criteria that need to be verified for a cITS under design or in execution.

The Principle of *Limited Retention* again complements the purpose specification and the consent by the user. If the service or operation has been fulfilled there does not seem to be any reason to keep this information any longer. However, as already indicated before there might exist legal or governmental regulations that require such information to be kept longer than for the fulfillment of the purpose. Therefore, additional technical measures are necessary that help to obey both (possibly conflicting) requirements.

The Principle of *Accuracy and Context Preservation* seems to be controversial at a first glance since accuracy of information is closely related to integrity and quality of the cITS overall. However, we argue that inaccurate information also impacts the data subject's privacy – or as stated in the APEC Privacy Framework [7] "Making decisions about individuals based on inaccurate, incomplete or out of date information may not be in the interests of individuals or organizations . . . ".

The Principle of *Security* requires technical measures that protect the data subject's information against ". . . loss or unauthorized access, destruction, use [for which there does not exist consent] modification, or disclosure of data." [6].

The Principles of *Openness* and *Compliance* pose particular challenges as the cITS is a distributed system where participating components might change over time. Therefore, specific technical measures might be necessary to fully comply with this principle for cITS. In particular, to verify compliance might be technically challenging as well as a costly task.

2.3 First Steps towards a Privacy aware Architecture

Before using the HcITS principles further, we first review some of the existing architectures that have been developed by other European Projects. This review together with the guiding HcITS Principles provides us with a solid basis to lay out the first steps towards an architecture for the PRECIOSA project. The following subsection presents several cITS architectures that are currently under discussion in various (European) projects. These architecture do not focus on privacy, however they provide a generic view on the functionality and the components of a cITS system. Based on the insight we gain, we propose a first generic view for a cITS system in PRECIOSA.

2.3.1 The C2C-CC Architecture

The CAR 2 CAR Communication Consortium (C2C CC) defines itself as *... a non-profit organization initiated by European vehicle manufacturers, which is open for suppliers, research organizations and other partners. The CAR 2 CAR Communication Consortium is dedicated to the objective of further increasing road traffic safety and efficiency by means of inter-vehicle communications. The objectives of the C2C CC are to create and establish an open European industry standard for C2C communication systems and to guarantee European-wide inter-vehicle operability, to enable the development of active safety applications by specifying, prototyping and demonstrating the C2C system, to push the harmonization of C2C communication standards worldwide, and to develop realistic deployment strategies and business models to speed-up the market penetration.*[2]

In its Manifesto [1], the C2C-CC presented a “draft reference model”, that gives an initial idea of an architecture for C2C systems. According to Figure 2.1, the architecture consists of various entities, networks, and interfaces, including those of C2C providers and telcos. Therefore it already anticipates the idea that vehicles will communicate among each other via DSRC as well as with backend systems via heterogeneous networks consisting of RSU equipment or cellular networks.

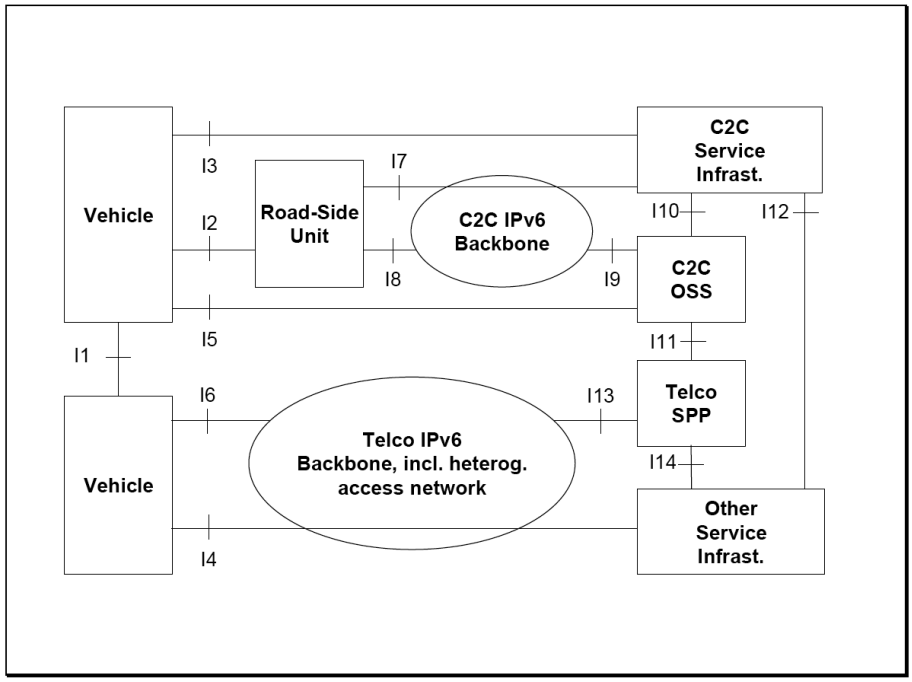

Figure 2.1: C2C CC Draft Reference Model (from [1])

[2] shortened from http://www.car-2-car.org/

2.3.2 The Harmonized European ITS Architecture

The idea of a harmonized European ITS architecture was also followed by the COMeSafety project. Its European ITS Communication Architecture [2] provides a baseline architecture framework for cooperative systems. It has been developed as a joint effort together with the EC funded projects COOPERS, CVIS, and SAFESPOT and in cooperation with the CAR 2 CAR Communication Consortium, ETSI, IETF, and ISO and with input from IEEE and SAE. It therefore represents a consensus of all major participants in this domain. It is currently regarded as an agreed base by many stakeholders. Based on contributions from the SeVeCom project, the architecture considers basic security and privacy aspects.

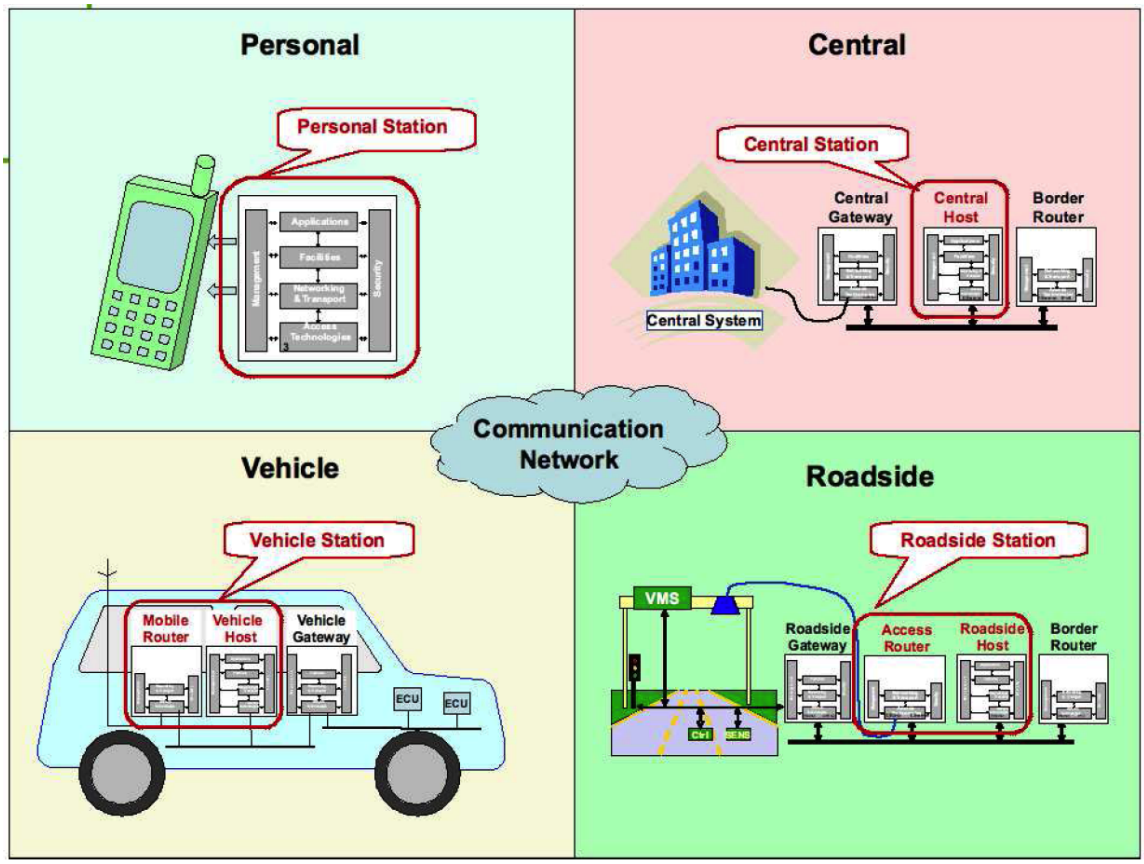

Figure 2.2: COMeSafety European ITS Architecture - Architecture Components (from [2])

Figure 2.2 shows the major components in the European ITS Communication Architecture. While not being as detailed in terms of the communication relationships as the C2C Reference Model, it provides a more detailed view of the internals of the nodes which is further detailed in Figure 2.3. The figures reflects the vision of COMeSafety of a software stack consisting of access technologies, networking, & and transport, facilities, and applications. Transversal components for security and management augment this stack. This

same stack is used in all kinds of nodes, be it personal devices, vehicles, central systems, or roadside units.

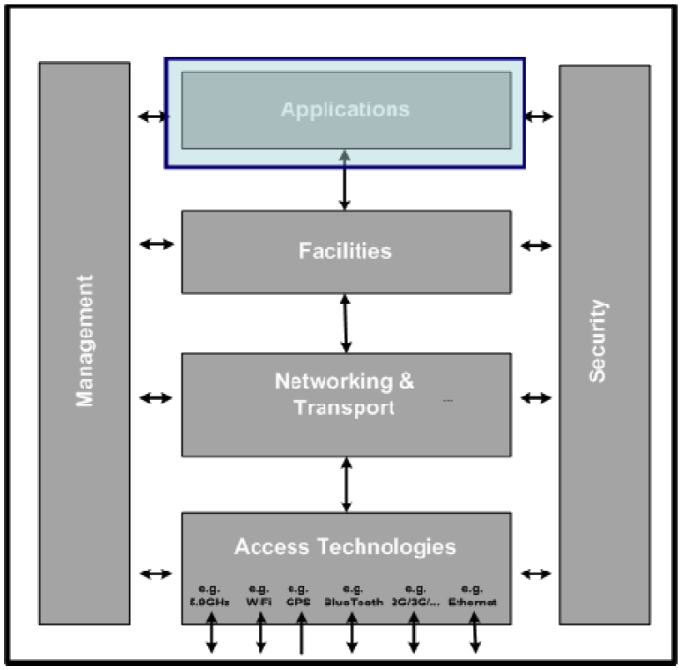

Figure 2.3: COMeSafety European ITS Architecture - Station Reference Architecture (from [2])

2.3.3 Simplified View

Based on the architectures introduced, we develop a simplified view on an cITS architecture as presented in Figure 2.4. This architecture will serve as a starting point for integrating privacy protection mechanisms in Chapter 5. Our initial approach abstracts from details of the communication system, as it was already done in the COMeSafety architecture. From a privacy point of view, it is primarily irrelevant what network connection or network protocol is used for communicating personal information. Sections 3.3 and 4.3 will present a layered reference model that allows a more fine-grained modeling as part of our design process when needed. Additionally, we see that we have subsumed the communication stack, the access technologies and facilities from the station reference architecture into a block called middleware and extended the model by sensors and data repositories, as they play a significant role as data sources and sinks in our later architecture.

Given this simple cITS model, we will now motivate how our HcITS principles will influence the design of our HcITS architecture.

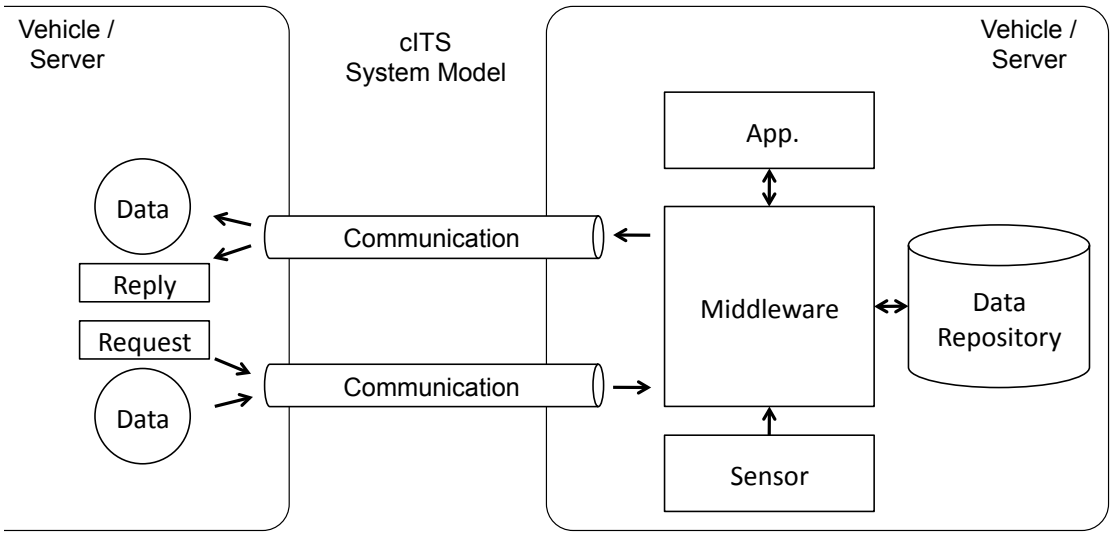

Figure 2.4: Simplified View on cITS Architecture

2.3.4 Embedding the Hippocratic Principles into the cITS Architecture

Figure 2.4 provides the solid basis to refine the cITS architecture by additional subcomponents such that those embed the ten principles of HcITS as introduced in Subsection 2.2. We realize that the architecture must allow us to include user privacy preferences in a flexible and dynamic manner. That is, the user should be able to change his/her preferences over time. In particular, the user must be able to specify (and to express) the purpose for processing his/her data (Principle 1) and to give consent (Principle 2).

For this reason, we use the concept of policies to specify those privacy preferences which must be known and processed by the extended middleware. We therefore include into the Middleware component

- A Privacy Policy Manager
- A Privacy Control Monitor
- A Privacy Trail Manager

To implement Limited Retention Principle (Principle 6) and to guarantee secure data storage and communication (Principle 7) we introduce

- A Data Retention Manager
- An Intrusion Detector

The refinement of the Middleware Component is shown in Figures 2.5. We notice that, we added a separate "off-line" tool named Data Collection Analyzer & Intrusion Model Generator. This subcomponent is responsible for generating models for intrusion detection

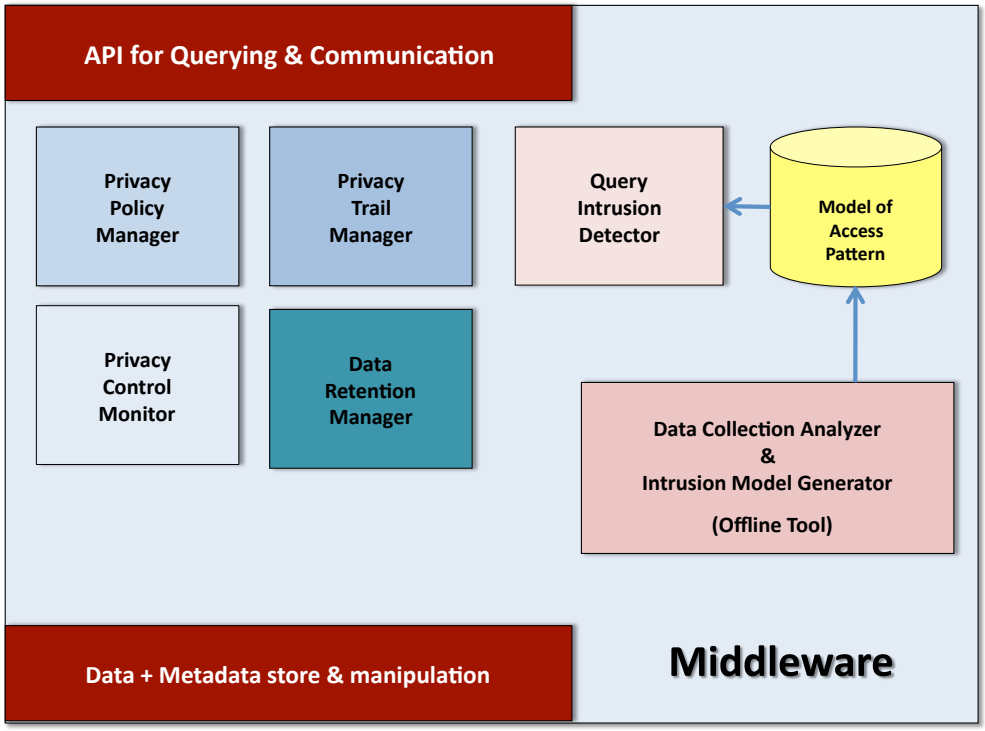

Figure 2.5: A generic Architecture for HcITS

at runtime similar to the component derived by Agrawal [4]. More details about those components will be discussed in Chapter 5.

Furthermore, we notice that the *Principles of Limited Collection*, *Limited Use*, *Limited Disclosure* and *Accuracy and Context Preservation* need a broader approach. To guarantee these principles, we must ensure that those are already checked during the design and implementation phase and during deployment phase of the system. They also influence other parts of the system, e.g. requiring confidential communication and an organizational framework controlling data access.

2.3.5 Beyond a HcITS Architecture

So far, we extended the cITS architecture with additional components to implement the privacy principles functionally. However, we also realize that additional extensions are necessary to fully ensure that the system obeys the privacy specifications provided by the user. Those are:

The use of metadata The Hippocratic principles clearly show that user data must be accompanied with additional data that go beyond determining possible domain values

or structural properties in order to be able to enforce privacy. In the context of HcITS, those kind of metadata must be extended to guarantee the proper access and dissemination of data within a cITS. A comprehensive schema design for all metadata necessary in an HcITS is central for successfully enforcing privacy policies and privacy preferences in an HcITS.

Obviously, the source of this kind of metadata should be the result of analyzing privacy preferences expressed by a privacy policy language and deriving the meta data from expressions of this language. We discuss aspects of such a policy language further in Section 4.2.

The definition of system privacy As cITS is a distributed system consisting of several components and an underlying (communication) network, it becomes especially important to understand how to build such a system from basic components. Similar to building correct distributed systems from correct components (independently of the definition of correctness), we must understand and ensure how to build HcITS from individual components that already exhibit privacy properties that are known. That is, the composition of a cITS from different components must include a clear process that derives and guarantees a level of privacy for the cITS based on the privacy properties of the individual components. Only such a composition approach guarantees that the overall system enjoy a verifiable level of privacy.

2.4 Summary and Outlook

This section provides the solid foundation for Hippocratic cooperative ITS systems by introducing implementation independent principles that should guide any future design and development process for such systems.

In today's systems the principles of *purpose specification*, *consent*, *compliance*, and *openness* are often addressed only implictly. Therefore, systems are built for a given purpose where the consent of the user is assumed by the system and compliance verification or checking of openness is often not supported. We envision a design process, where processing those information is an inherent design feature of the system to be built and therefore needs to be considered as explicit requirements.

The principles of *limited collection*, *limited use*, *limited disclosure*, and *limited retention* are to be considered already during design, meaning that the design process should assist a designer in finding that system architecture that best combines the desired functionality with those constraints. Finally, *accuracy and context preservation* and *security* need to be modeled as explicit requirements that the system must consider.

Furthermore, we also motivated the need for describing data (metadata) are necessary to enforce privacy policies as specified by the user, and for the concept of system privacy to guarantee that privacy specifications are not only handled correctly by each component in a distributed system, but by the overall system as well. We shall discuss these aspects more in later chapters of this document.

Therefore, we investigate the following important aspects even further :

- a detailed description which metadata are needed to maintain a HcITS;
- when to enforce privacy policies during the different phases of the life cycle of a cITS;
- how to maintain an overall privacy level (which we call system privacy) based on the assumed and specified privacy specifications for each component in a cITS.

A more detailed, design and implementation based discussion about the necessary technical properties is left to later Chapters and deliverable D10.

3 cITS Design Process - Architecture Framework

As part of the general architecture framework, this chapter outlines the PRECIOSA design process for designing privacy-aware and privacy-verifiable ITS systems. The purpose of the architecture framework is to provide general guidelines and processes for designing arbitrary privacy-aware architectures, to fulfill this goal the discussion of the design process is kept on an abstract level. This way, it can also be adapted to develop privacy-verifiable architectures for other domains.

3.1 Design Process Methodology

The design process of the cITS architecture framework is founded on the requirements and principles discussed in Chapter 2. It facilitates the design and development of privacy-verifiable architectures.

Hereby, we distinguish between design time and runtime and incorporate both into the architecture framework. At design time, it is important that the design process, which is described in this chapter, is privacy aware. However, this does not only mean that privacy should be considered from the beginning of an architecture design but also that verifiability of privacy is taken into account. Addressing privacy verifiability at design time facilitates the assessment of the level of privacy provided by a system architecture at runtime. The distinction of design time and runtime also enables the development of distinct processes that may be applicable for only design time, only runtime, or both. Hence, the distinction of design time and runtime is a recurring and important aspect of the architecture framework.

3.1.1 Guidelines for Privacy-Aware Applications

While it is highly important to provide a thorough design process for privacy-aware architectures, it is also an aim of PRECIOSA to derive guidelines for the design and development of privacy-aware applications. These guidelines provide application designers and developers with a concise set of rules that can be followed to enhance the privacy friendliness of their applications.

In the previous chapter, preliminary design guidelines have already been formulated but they are still quite general. They must be refined and structured in various ways:

- The guidelines must address and support all phases of the life cycle by extensions and refinements to create privacy aware systems. These guidelines must take into account potential threats to privacy, available privacy-enhancing technologies, and also concepts to measure privacy such as k-anonymity, l-diversity and others.
- Furthermore, we also envision guidelines of less technical matters. Such guidelines could help to promote privacy aware systems focusing on organizational or legal matters in a company organization. They could also support the awareness and sensitivity for privacy and privacy aware systems.
- The guidelines should also help users to express their privacy needs and expectations using a given policy language.

These guidelines will be derived from the design process which is presented in the following and reflect legal regulations as well. Thus, we shall further focus on these various aspects in deliverable D11 “Guidelines for privacy aware cooperative application”.

3.1.2 Structure of Design Methodology

To design privacy aware systems we suggest a methodology consisting of four steps as shown in Figure 3.1. We first perform a system design using currently available approaches. Separately, we identify the different needs for privacy protection using an appropriate policy language as discussed in the next section. In a third step, those requirements must be mapped into the (technical) mechanisms provided by the underlying design and runtime environment. In a last step the different mechanisms and components must be combined to result in the specified system. How to perform the composition will be discussed in a later deliverable. It should also be noted that the whole design process potentially needs to be traversed multiple times, as identified privacy requirements and mechanisms might require changes to the system model. Additionally, for the Information Flow Analysis we propose to design different alternatives for comparison in any case.

We adapt this basic design methodology for privacy aware systems

1. by using various metamodels and schemata to guide the overall system design (Step 1);
2. by refining step 2 as follows (see Figure 3.2)
    - create or adapt an adversary model that describes what kind of attacks should be taken care of;
    - use this adversary model to identify points of control and observation (PCOs) for privacy protection, as described in Section 3.4;
    - based on the identified PCOs describe the kind of operation and the kind of measures necessary to protect privacy;
    - based on the selected measure derive a value that reflects (measures) the level of privacy protection.

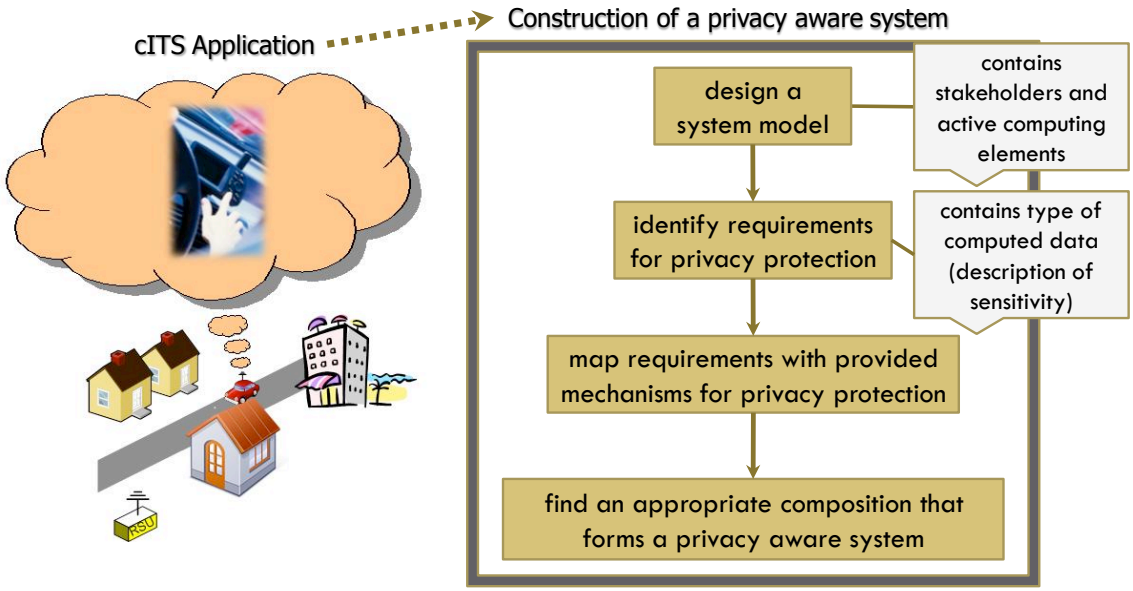

Figure 3.1: Methodology for designing a privacy aware system

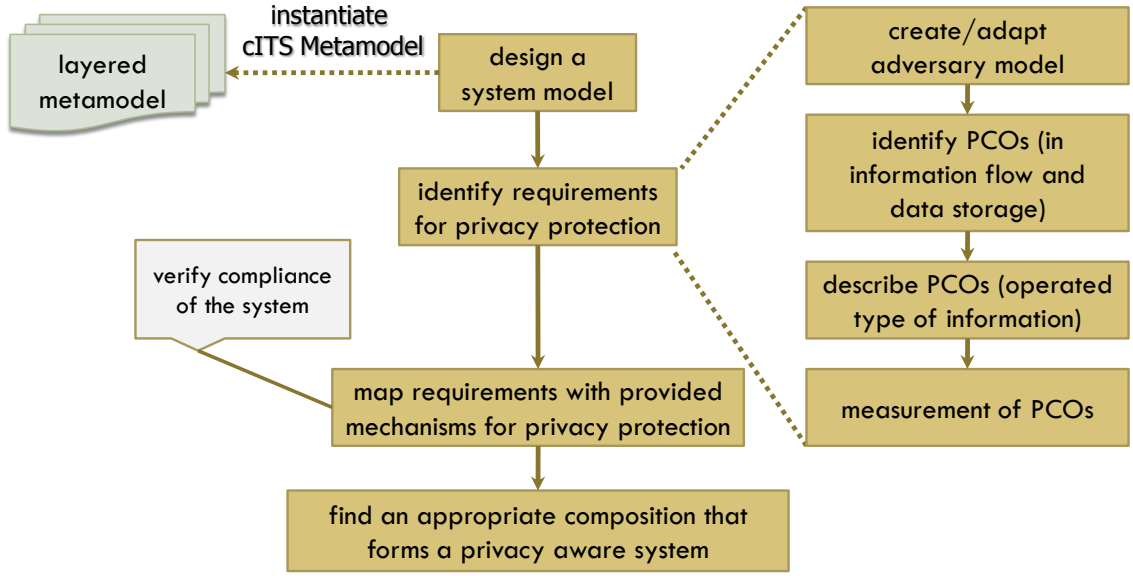

Figure 3.2: Methodology for designing a privacy aware system - PCOs

Figure 3.3 slightly varies the previous approach for applying similar steps to existing systems. Here the goal is to verify if the requirements derived in the first two steps match the existing system. Of course, this step might be undecidable in general; therefore it is necessary to understand how far such an verification methodology will help to detect mismatches.

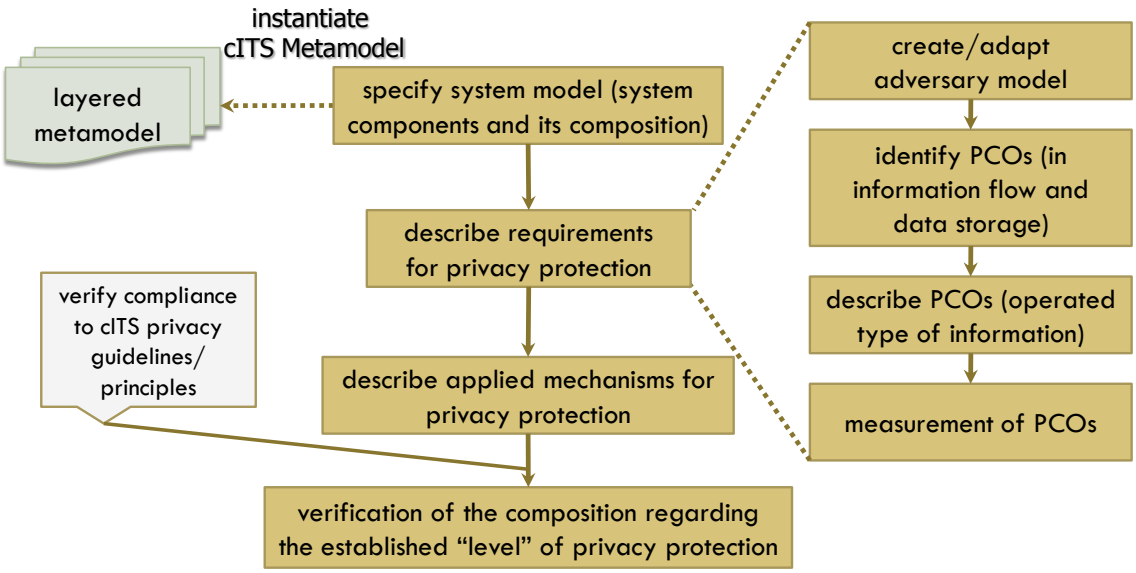

Figure 3.3: Methodology for verifying an existing system regarding privacy

3.2 Policy Design

Specifying the requirements regarding privacy is one major aspect for a privacy aware architecture. These requirements are specified by users (data subjects), service providers, providers for privacy aware components and systems, public authorities, law, data controller and others. Measurements of the utilization of systems, components, and applications regarding privacy are always based on specified privacy requirements. Thus the description of privacy requirements have to be unambiguous. Thus the use of well defined (standardized) metamodels and models is necessary. To support the creation and evaluation/verification of such model based descriptions ontologies can be used to check for inconsistencies and contradictions. Furthermore, standardized vocabulary description languages like the Web Ontology Language (OWL) are powerful enough to express a big variety of terms and conditions.

Existing approaches to describe privacy requirements (like P3P) are mostly not flexible enough to deal with changing situations. Specified rules cannot be used to capture new situations that are similar to ones previously defined. For instance syntactical differences like the usage of synonyms or specializations of situations are not addressed. With the use of machine processable descriptions in OWL we are able to infer new situations and to apply previously defined rules to the inferred ones. New challenges can easily be addressed by just updating the ontology descriptions.

Privacy policies can be specified and published both at design time and at runtime. At design time, we have to specify the expressiveness of such a policy language. As mentioned above the privacy requirements have to be based on a well defined metamodel.

Such a metamodel has to define clearly the terms and its relationships used to express requirements.

In deliverable D6 we define metamodels for the domain of privacy in cooperative ITS. Thus a policy language for the HcITS architecture can use this metamodel to adequately express the privacy requirements. Evaluating system policies (for instance derived from the principles) at design time is not challenging. At runtime the evaluation has to be just in time. Therefore it is not possible to use a logic like OWL Full because in general with OWL Full we can express statements that are not decidable, and thus may not terminate. The challenge is to define a policy language that is expressive and flexible enough to capture all necessary privacy requirements and at same time can be efficiently evaluated. To ease the definition of appropriate policy languages we investigate in deliverable D6 the use of a meta policy language and exemplary define a policy language for our prototype of the HcITS architecture.

Translating “abstract” privacy policies to concrete usage policies can be one approach to deal with the complexity of online inference and the complexity of compliance check for the specified privacy policies. The main idea is to perform extensive computations like inference offline or in parallel and to use simple access patterns for the access right management at runtime. For instance the process of updating the ontologies or the inference of new situations does not directly influence the running system. One main challenge of this approach is to update the access policies periodically and to correctly translate the privacy policies. As a result we derive the access model of applications/services from the specified privacy policies. Another challenge is the handling of privacy policies that do not directly translate to access policies, but require modification of stored data, e.g. limited retention of data items or the decrease of detail of information after a certain time.

Most of the principles can be expressed by creating policies. For instance, limited retention can be defined in general, so that data will never be stored longer than two years or, in special cases, the data will be deleted after 1 minute. Depending on the sensitivity of data there are different requirements on how to protect information. For instance, for some type of information it might be necessary to guaranty a high level of anonymity while for other information it is important that it is never combined with another special type of information resulting in the identification of individuals.

Furthermore, the requirement for privacy depends on the type of operations that are performed on the information. For example, there could be privacy issues regarding the communication of information, while a secure data store may ensure the protection of privacy-relevant data, thus it might not matter at what detail-level information is stored because the storage is trusted. On the other hand, it could also be that a secure communication channel protects privacy-relevant data, but that only insecure data storage is available which in turn requires that data is stored in a privacy aware manner.

The example in Figure 3.4 demonstrates the definition of a policy describing a situation where it is forbidden that two abstract components (component 3 - c3 and component 4 - c4) receiving some special kind of information (s12 - not sensitive, s13 - not sensitive) are realized/implemented by one instance. In the example the reason for this restriction results from the possible inference of personal information from the received information of both

components. For identifying such relationships deliverable D6 introduces a model that can be used to describe (a) personal information, (b) identifiers, (c) the inference of information and identifiers, and (d) the connection from identifiers with personal information resulting in personal information.

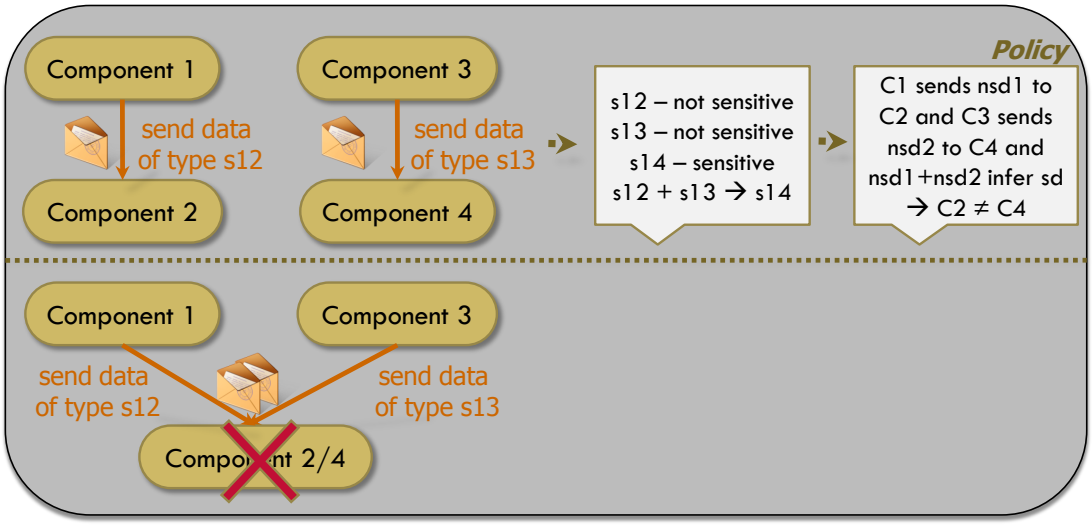

Figure 3.4: Policy that forbids the combination of information in one component

3.3 Layered Reference Model

In order to analyze ITS during the design process, we need to build a model. The basic ideas of this model have already been outlined in deliverable D4 [8]. In this deliverable we will describe the model's expressiveness to allow for later investigation of privacy relevant points in the system design.

We decided to use a graphical model to allow the designer to visualize the ITS in order to gain a deeper understanding of the information flows within the system. One important goal of this modeling is to be able to compare various designs of an ITS. For example, data processing could happen more centrally in the backend or in a distributed manner closer to or on the mobile nodes. This has significant privacy implications as in the first case data is centrally collected whereas in the second case data is dispersed in the system and nodes only see a small subset or aggregated data. By visually comparing the models of different design alternatives, the system designer gets an intuitive idea of their advantages and disadvantages in terms of privacy protection. In addition, the information flow layer even allows a kind of privacy measurement to compare the level of privacy provided by different system designs as discussed in Section 3.4.

As it is virtually impossible to combine all different aspects of such a model into a single figure we decided that our model supports multiple *layers*, or *views*, that each describe different aspects of the same ITS.

A given ITS scenario is modeled in different layers or *views*, where each view highlights certain aspects of the system. Those models will then serve as the basis for a privacy analysis.

- **Entity View:** The entity view shows all participating entities and their relationships or interfaces. Entities can be devices, persons, or organizations. A link between two entities symbolizes a direct exchange of information between those entities during system operation. Entity views allow for a quick overview over participating entities without detailing the nature of their interaction.
- **Deployment View:** The deployment view describes available ICT devices in the ITS that can be used for deployment of system functionality. These can be road-side units or on-board units, but also servers that system operators use in the backend. Links between devices represent communication channels over which information can be exchanged. The deployment view becomes especially important for PRECIOSA as the same functionality within an application can often be deployed in different parts of the system being under the control of different stakeholders. This deployment decision then controls where personal information is stored or sent to and which stakeholders have access to it.
- **Logical Function View:** The logical function view splits the overall application into different functional units and models the relations and interfaces between those units. Links between functions represent interfaces where information is exchanged. Interfaces can be APIs, network protocols, etc.
- **Information Flow View:** The information flow view is based on the logical function view, but emphasizes the exchange of data between entities. It is somewhat comparable to dataflow diagrams [9] as they are used in the Structured Systems Analysis and Design Method (SSADM) [10] or the activity diagrams in UML [11], however, we have adapted the graphical syntax to give a better expressiveness for our goal of visualizing the flow of personal information in an ITS system.

The different views can be described using diagram types that we already introduced in Deliverable D4. Focusing the descriptions on the ITS domain we identified the three specific subdomains vehicle domain, access domain, and backend domain. Each of the domains describes parts of the systems which are logically separated by system boundaries.

The *Vehicle Domain* contains all elements that are located in the vehicle(s) and are thus mobile and legally or physically belong to the driver(s) or other passengers. All elements that have the primary function of enabling communication between the vehicle domain and the backend domain belong to the *Access Domain*. Finally, the *Backend Domain* groups elements that are centralized and store or process data provided by the other elements.

Combining the different views allows a detailed analysis of the architecture of an ITS application and the interaction of its parts. In the next section, we describe how especially the information flow view can be used to analyze the privacy issues and protection of privacy in an ITS system design. Section 4.3 describes the graphical syntax of our model and provides some examples. Finally, Section 4.4 demonstrates the information flow analysis.

3.4 Privacy Analysis of Information Flows

When modeling an ITS with this layered reference model, one may identify points of control and observation (PCOs). PCOs are specific points in the information flow of a certain application use case where the level of privacy provided by the system should be measured and verified.

In general, privacy effects should be analyzed when information is being processed as well as when information is flowing from one entity to another and especially when information crosses domain boundaries. The reasons why privacy has to be verified when information is being processed and when information from different sources is being combined is obvious – additional information and data fusion may alter the level of privacy at this point. Analyzing the level of privacy when information leaves an entity is important for determining the effectiveness of applied privacy mechanisms. Analyzing the level of privacy when information crosses domain boundaries is important because once information reaches a different domain more information may be available it could be combined with, thus reducing the provided level of privacy, e.g., when service operator in the backend generate large centralized databases containing data about large populations of users.

PCOs at entity and domain boundaries can be identified based on the entity and deployment views. However, the logical function view, and especially the information flow view, provide more fine grained information about the amount and kind of information being transfered throughout the system. This enables to determine more specific PCOs at critical points in the system, e.g., when information from different sources is combined or processed together.

Additionally, to ensure that privacy is verified at all crucial points in the modeled information flow, potential attacks on privacy can be identified to determine further PCOs. Several methodologies for attack analysis and risk assessment exist that can be utilized for this purpose. Most of them require a prior definition of an adversary model to estimate the capabilities of a potential adversary. The attack analysis should be based on the modeled information flow, and facilitates the identification of PCOs other than those at entity and domain boundaries.

Now, the basic concept of *privacy analysis of information flows* is to model various alternatives of system architectures using the layered reference model, and use the recommended metrics described in deliverable D2 “V2X measurement approach” [12] to quantify the provided level of privacy via the identified PCOs. Of course only metrics that can be applied during design time are applicable here. Alternatively, real-world tests need to provide at least statistically significant estimates. Applying this approach then allows a qualified

selection of the most privacy-friendly system design. Backes et al. [13] demonstrate the possibilities of the analysis of information flows on a more theoretical level but restricted to the information flow of locally executed source code using a line-by-line analysis.

In summary, the following steps are required for a full privacy analysis of information flows:

1. Model system architecture using layered reference model (entity, deployment, logical function, information flow views).
2. Identify attacks based on the modeled views and previously created adversary model.
3. Identify PCOs based on modeled views and identified attacks.
4. Evaluate PCOs and measure privacy level of system design with privacy metrics.
5. Repeat steps 1 to 4 with alternative system designs.
6. Use the results to either select the best candidate for implementation or to develop more alternatives.

4 cITS Design Process - Reference Architecture

4.1 Methodology-Guided Design and Implementation Process

The design and implementation process of privacy-verifiable cITS applications is based on the process discussed in the architecture framework (see Section 3.1). This general approach needs to be integrated with existing software design methodologies, e.g., the classical waterfall model or the Rational Unified Process (RUP) [14]. During the software design process of an ITS, privacy aspects need to be taken into consideration at various stages. We will use the waterfall model only as an example and discuss the privacy-related activities during the different phases:

Requirements specification: The requirements should also include privacy requirements deduced from the privacy guidelines, i.e., it should be described what personal information is processed by the system, what are the specific protection requirements, what stakeholders and entities are involved, and so forth.

Design: During the design phase, the application or different alternative application designs should be modeled using e.g. the layered reference model described in Section 4.3. Alternatively, comparable approaches from model driven development can be used, for instance activity diagrams, data flows, and requirements engineering mechanisms. Independent of the approach used, an information flow analysis is carried out for each alternative (see Section 4.4). This allows a qualified selection of the best alternative from a privacy point of view. Design should also respect the privacy guidelines described in deliverable D11 “Guidelines for privacy aware cooperative application” which includes implementing the PeRA architecture (see Chapter 6).

Implementation and Integration: During implementation and integration, compliance with guidelines and the design has to be monitored.

Validation: During validation, a trusted third party needs to verify and certify compliance with guidelines and correct implementation of the PeRA architecture.

Installation: Installation also includes setting and configuration of privacy policies with respect to user preferences. Albeit policies should be adaptable and users should be able to easily customize them to their needs and wishes, reasonable defaults tuned to the initial interests of individual users at installation and deployment time will safe users the effort of making big changes later on.

Maintenance: Maintenance and regular operation includes re-certification of the system and regular audits using the audit component to ensure compliance of the system.

As previously stated, these steps need to be integrated with the system design process that is in use in the specific organization implementing an ITS. There will also be the need to implement tool support to efficiently include guideline verification with design and testing procedures. This will also be covered in deliverable D13 “V2X privacy mechanisms and verification support tools”.

In addition to extending the classical software development cycle, we must also address the challenge of how application developers are able to design privacy aware ITS applications that make use of the features of the PRECIOSA architecture framework (design process and run-time architecture). As possible starting points we envisage the following two approaches:

1. In the short run we envisage a concerted approach (e.g., discussion in application security groups and with data protection agencies) to describe generally agreed privacy profiles. Those consist of:
   - policy profiles, and associated assurance and verification approaches (following the PRECIOSA verifiable architecture approach);
   - rules and methods for integrating privacy requirements into application design, generation, and application deployment.
2. Define a privacy aware ITS design discipline. This could mean that privacy and privacy policies can be designed at an early stage and then taken into account at each step of the design of an ITS application. PRECIOSA is currently investigating scenarios based on model-driven engineering, whereby a privacy metamodel is defined, instantiated into application specific privacy models, further instantiated in privacy friendly ITS applications.

4.2 Policy Design

As already discussed in Section 2.2 we must provide the user with a language that allows her to express her preferences to maintain the desired privacy level. In the following we do not describe a detailed language but rather discuss certain guiding principles for designing (or choosing) a policy language for expressing privacy preferences. Such a language should satisfy the following requirements

1. to express or to handle all aspects of all Hippocratic principles as introduced in Subsection 2.2.2;
2. to be simple enough such that the user can express her/his privacy preferences without being a privacy expert;
3. to be powerful enough such that experts are able to determine their individual privacy preferences on a fine grained, technical level.

Since the latter two requirements seem to be contradictory we envisage a policy language with (at least) two sublanguages: one for the novice, non-expert user, and one for expert users. Rather than keeping both languages disjoint we envisage to map the former language into the latter before generating the necessary metadata (as outline in the example of Section 3.2 from those expression. The leveled approach is shown in Figure 4.1. Details on the design of the policy language will be given in deliverables D6 and D10.

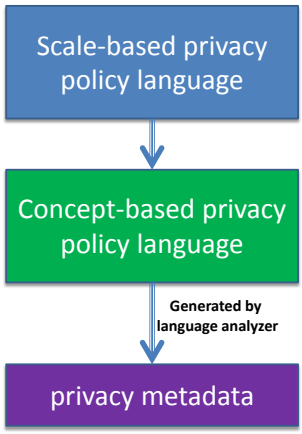

Figure 4.1: Two-level privacy policy

4.3 Layered Reference Model

While Section 3.3 described the more abstract model applicable to cooperative ITS in general, we now define a specific layered reference model for the design of privacy-aware ITS applications.

In deliverable D1 [5] we already did some analysis of privacy aspects/issues. Now we extend the analysis and introduce additional mechanisms to describe the applications and to identify its privacy issues and appropriate requirements for protecting privacy. For this purpose, we develop a graphical syntax based on the early ideas presented in deliverable D4 [8], that can be used to model the entity view, deployment view, logical function view, and information flow views of the layered meta model.

Each view has different semantics that have to be reflected in the modeling syntax. Therefore, we first describe general modeling aspects applying to all views, before separately describing the syntax for each view. Thereby, the syntax for the different views are kept coherent to ensure consistency. To facilitate better understanding of the presented syntax, for each view the syntax description is followed by a specific example. The floating car data use case outlined in deliverable D1 [5] will serve as an example and will be modeled with all views of the layered reference model in the following sections.

4.3.1 General Modeling Aspects

Each view has to take the three domains – vehicle domain, access domain, and backend domain – into account. All diagrams have to clearly visualize the boundaries of these domains to indicate the most important system boundaries. The purpose of the domains and what is contained in them has already been discussed in Section 3.3. In the view diagrams, domains are represented by markers at the bottom of the diagram. All three domains have to be present in every diagram. If one of the domains is not relevant for the diagram at hand it cannot be omitted but reduced in size. The reason for this is to ensure comparability between diagrams modeling different architecture alternatives for the same application.

4.3.2 Entity View Modeling Syntax

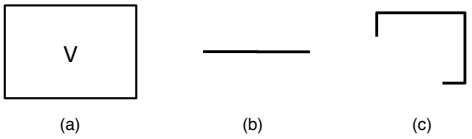

Figure 4.2: Syntax elements of the entity view: entity (a), relationship (b), self-referential relationship (c).

The entity view is concerned with all entities that are part of a system and their relations. Entities are represented as rectangular boxes (Fig. 4.2a). A box can represent a single instance of an entity or a class of entities if several entities of the same kind are participating in the system, e.g., a number of vehicles would be represented by a single entity box.

Relations between entities are understood as interfaces or communication links that symbolize direct exchange of information between these entities. The entity view does not distinguish between uni- and bidirectional information exchange, therefore relations are expressed by undirected connecting lines between entities (Fig. 4.2b). Relations between entities of the same class, e.g., communication between two vehicles, can be modeled with an undirected line starting and ending at the same entity (Fig. 4.2c). Interfaces and connections are not labeled, but can be easily referenced by the two entities they connect, e.g., $V - NP$ refers to the link between a vehicle and the network provider. Such references are unique because each relationship only connects two entities.

The entity view provides a quick and high level overview over participating entities without describing interactions between them in detail.

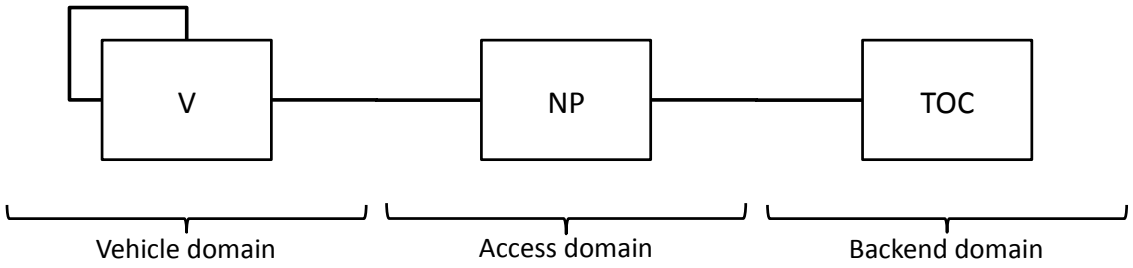

Figure 4.3: Entity view of the floating car data use case

Example: Floating Car Data

Figure 4.3 depicts the modeled entity view of the floating car data use case. Three entities are involved: vehicles (V), a traffic operation center (TOC), and a network provider (NP) that connects the vehicles and the TOC. Vehicles may also exchange information on traffic status among each other via inter-vehicle communication ($V - V$).

4.3.3 Deployment View Modeling Syntax

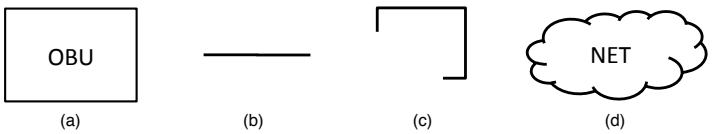

Figure 4.4: Syntax elements of the deployment view: device (a), communication channel (b), self-referential communication channel (c), opaque system parts (d).

The deployment view provides a more detailed and more technical view on available ICT devices in the ITS that can be used for deployment of system functionality, e.g., RSUs, OBUs, or servers. Similar to entities in the entity view, devices are represented by rectangular boxes (Fig. 4.4a). Links between devices (Fig. 4.4b) represent communication channels available for communication exchange and can again be self-referential (Fig. 4.4c), e.g., for V2V communication. Communication channels can be referenced by the two devices or deployment points they connect, e.g., $OBU - RSU$ references a communication channel between a vehicle OBU and a roadside unit.

Opaque parts of a system can be symbolized by a cloud (Fig. 4.4d). This may be required for parts that are not under the control of the system designer, e.g., devices in an access network provided by a network provider are opaque and can therefore not be utilized for deployment.

Example: Floating Car Data

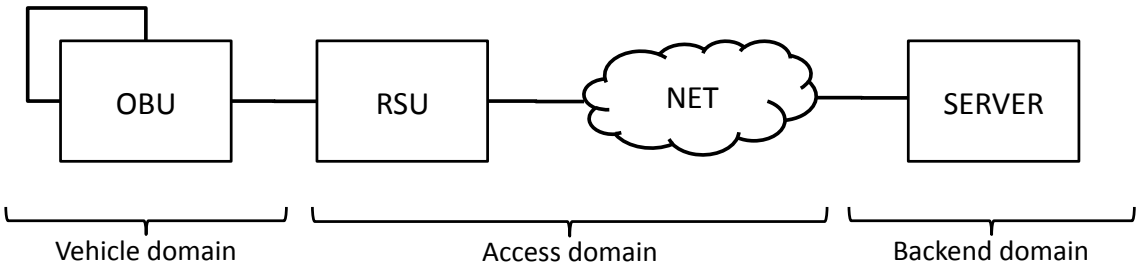

Figure 4.5: Deployment view of the floating car data use case

The deployment view for floating car data, depicted in Figure 4.7, includes OBUs (OBU) in vehicles communicating with other vehicles ($OBU-OBU$) or roadside units (RSU). RSUs are connected to an opaque access network (NET). This network provides a connection to the TOC server ($SERVER$) in the backend.

4.3.4 Logical Function View Modeling Syntax

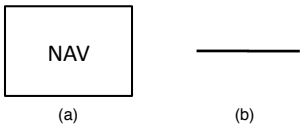

Figure 4.6: Syntax elements of the logical function view: functional unit (a), interface (b).

The logical function view models an application on the system development level by splitting the overall application into different functional units and modeling the relations and interfaces between them. Functional units are represented as rectangular boxes (Fig. 4.6a). Interfaces between functional units are represented as undirected connecting lines (Fig. 4.6b) that can be referenced in the same way as for the entity and deployment views.

Example: Floating Car Data

Figure 4.7 depicts the logical function view of the FCD use case. Two main functions are supported: vehicles report probes of road status data to a traffic operation center (TOC) and vehicles consume up-to-date road status information from the TOC.

For the data probes to be sent to the TOC, vehicles determine their location using a localization or positioning service (LOC), data about the current road and traffic status has

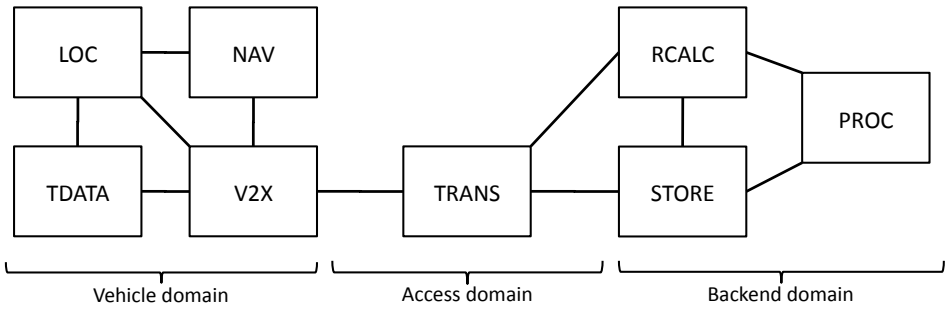

Figure 4.7: Logical function view of the floating car data use case

to be gathered and merged into a report ($TDATA$) which may also require localization (LOC). Via a V2X communication unit ($V2X$) the data is sent out and reaches the TOC server by means of a transportation service ($TRANS$), which abstracts from specific communication and transportation protocols. At the TOC server data is stored in a database ($STORE$). The individual probes are processed and analyzed together ($PROC$) to determine the current status of the road network. The results of the analysis may also be stored again.

Subsequently, route planning requests from vehicles can be processed and corresponding routing information can be computed ($RCALC$). This information can then be returned to vehicles whose navigation system (NAV) can take into account the received information for optimal routing decisions. Navigation (NAV) obviously also requires position information (LOC).

4.3.5 Information Flow View Modeling Syntax

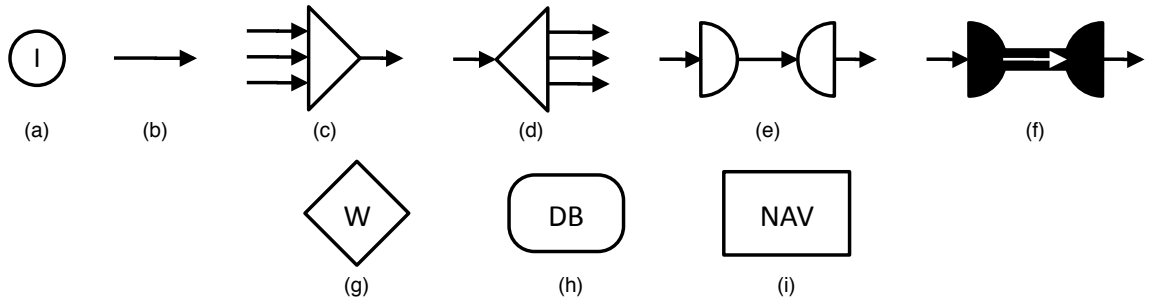

Figure 4.8: Syntax elements of the information flow view: data item (a), information flow (b), multiplexing data items (c), demultiplexing data items (d), send and receive (e), secure communication (f), information processing (g), data storage (h), information consumer (i).

The information flow view models the flow of personal information of an application. Data items that are newly created are depicted by a circle (Fig. 4.8a). Information flow is symbolized by unidirectional arrows (Fig. 4.8b). The combination, or multiplexing, of several data items into one message is symbolized with a triangle (Fig. 4.8c), hereby the single information items are only grouped together but not modified. Similarly, a message can be ungrouped, or demultiplexed, again (Fig. 4.8c).

Sending and receiving of information are depicted by a closing and an opening half circle, respectively (Fig. 4.8e). Thus, the internal information flow of an entity is always enclosed in these symbols. In case multiple entities of the same kind may participate in the application, a flow arrow can point from the sending symbol back to the receiving symbol of the same entity.

To symbolize a secure communication channel the symbols for sending and receiving information are filled, and the connection between them is represented by a black tube with a white information flow arrow (Fig. 4.8f). For secure communication channels it is assumed that the security holds from end to end, therefore intermediate nodes that forward encrypted data can be omitted.

Information processing, fusion, and aggregation are described with a diamond shaped symbol (Fig. 4.8g). Note, that information processing always results in a new data item that has to be modeled accordingly.

Information may also be stored, data storage is represented by a rectangle with rounded corners (Fig. 4.8h). Mere caching of information has to be modeled the same way. Finally, information consumers, e.g., a navigation system or a warning display, are symbolized by rectangles with pointed corners (Fig. 4.8i).

The graphical syntax for information flow modeling is clearly more expressive than the syntax of the previous views. Nevertheless, the syntax is only comprised of the items necessary to fully model information flow of arbitrary applications while retaining clarity of resulting diagrams.

Example: Floating Car Data

Figure 4.9 shows the modeled information view of the FCD use case. The Figure models information flow for reporting of traffic status data as well as for requesting and receiving routing information.

When contributing floating vehicle information, vehicles combine their vehicle identifier (I), the vehicle's current position (P), and traffic or road status data (T), e.g., information about traffic flow or icy road surfaces, into a message. Such a message may either be forwarded (and potentially cached) by a number of vehicles or directly reach the access domain, e.g., by a RSU receiving the message. In the access domain, messages are forwarded and potentially cached by an unknown number of intermediate nodes. When floating car data reaches the traffic operation center in the backend domain, it is demultiplexed and stored in the TOC's database (DB) for later processing and data fusion. Note that caching of

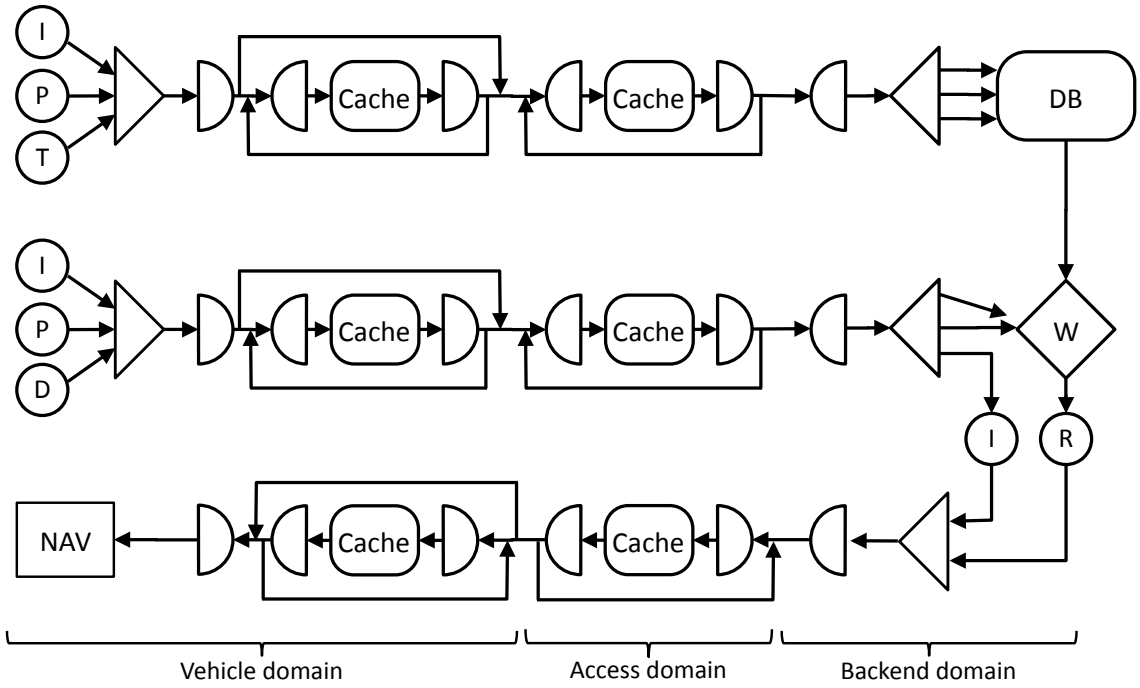

Figure 4.9: Information flow view of the floating car data use case

data can happen both in vehicle and access domain, thus it has to be modeled twice to account for potentially different characteristics of these domains (see Section 4.4.2).

For the consumption of floating car data, the individual probes stored in the database (DB) are analyzed and fused with each other and other information to model the current real-world traffic situation (W). Vehicles can request traffic information from the TOC by sending a request consisting of their identifier (I), the current position (P), and their destination or direction (D). When receiving such a request, the TOC processes P and D together with information from the database (DB) resulting in some routing information relevant for the vehicle (R). R and I are combined to a message and returned to the vehicle via the access network. The FCD client of the vehicle's navigation system (NAV) consumes the received information by calculating the optimal route to D with respect to the current traffic situation.

4.4 Information Flow Analysis

The basic idea of information flow analysis has been introduced in Chapter 3. We use the layered reference model defined in the previous Section (Section 4.3) to model a proposed system architecture or application. We then identify potential privacy attacks based on the modeled views and an adversary model. The knowledge gained this way facilitates

the identification of points of control and observation (PCO) in the information flow view, which are subsequently evaluated to assess the privacy level provided by the underlying system design. Afterwards, the process can be repeated for alternative system designs for the same application to compare the impact of different designs on privacy. Such an alternative could, for example, be the shifting of a logical function onto a different device based on deployment and logical function views.

In the following, we will discuss each of these steps in detail, thus outlining a methodology for privacy analysis of information flows that can be applied to arbitrary applications. As in Section 4.3, we use the floating car data use case as an example to demonstrate the process of information flow analysis.

4.4.1 Modeling a System Design

The first step of information flow analysis is modeling a system design for a specific architecture. Utilizing the layered reference model of Section 4.3, derived from the layered meta model introduced in Section 3.3, a system design can be modeled in several views. These views all describe the same design but at different layers of abstraction.

As the name implies, we are mostly interested in the information flow view for information flow analysis. However, entity view, deployment view, and logical function view are also important because they facilitate the accurate modeling of information flow and provide useful clues for modeling alternative, more privacy friendly system designs. Therefore, all views are modeled in this step.

For the specific modeling syntax of each of the four views we refer to Section 4.3 and its Subsections.

Example: Floating Car Data

The modeling of a system design with the layered reference model has already been exemplified in Section 4.3 with the FCD use case. Figure 4.3 shows the entity view of FCD, Figure 4.5 shows the deployment view of FCD, Figure 4.7 shows the logical function view of FCD, and Figure 4.9 shows the information flow view of FCD. It is the latter one we will be mostly referring to in the following.

4.4.2 Adversary Model

Before the information flow of an architecture can be analyzed for attacks an adversary model has to be defined that describes the assumed capabilities of a potential attacker. The capabilities and exhibited characteristics of an adversary depend on the chosen kind of attack. In Chapter 3 we identified three interlinked domains that are part of Intelligent Transport Systems:

- **Vehicle domain.** In-vehicle communication and inter-vehicle communication, also including data stored in vehicles (e.g., communication with mobile devices and other vehicles).
- **Access domain.** All means and devices that provide communication and caching between vehicle and backend domain (e.g., RSUs or proxy servers).
- **Backend domain.** Storage and processing of data provided by the other domains, as well as communication between backend servers and services (e.g., database servers or service providers).

Each of these domains includes several potential points of attack for an adversary and analyzed independently. This way, a specific adversary model for each domain can be compiled that well reflects the required capabilities of an adversary. Depending on the domain, an adversary may require more extensive capabilities to be able to mount a meaningful attack, which may also result in different costs associated with an attack. Domain boundaries are potential attack points as well. If the adversary models of two adjacent domains diverge the *stronger*[1] adversary model should be assumed for the boundary as well. At the same time, it is also important to have a consistent adversary model for the whole cooperative ITS. Therefore, the same methodology is chosen for all three domains to model a potential adversary. This allows us to either focus on domain-specific adversary models or generalize to a system-wide adversary model.

In this example, we use the adversary model the classification approach of Raya and Hubaux [15] as the basis. They classify adversaries along the dimensions *membership*, *method*, *scope*, and *motive*. We reduce the adversary model to the first three dimensions, because in terms of privacy an adversary's motive is irrelevant: regardless if an adversary is acting with malicious or self-benefiting intent, user privacy will be breached by a successful attack either way. The dimensions for the adversary model are therefore:

- **Membership.** An adversary can be an *insider* or an *outsider*. An insider is a valid member of the communication network, e.g., a vehicle, and possesses valid authentication credentials and other inside knowledge. An outsider is not part of the network and cannot take part in authenticated communication.
- **Method.** An adversary can be either *active* or *passive*. An adversary is actively taking part in communication when sending or retransmitting messages. A deliberate intrusion attempt into a system component is also an active attack. An adversary that only eavesdrops on communication or silently gathers information on communication partners is acting passively.
- **Scope.** An adversary can have *local* or *extended* scope. With local scope an adversary's capabilities are limited in reach, so that only certain parts of the system can be attacked at once, i.e., an adversary can mount an attack against one backend server, or eavesdrop on vehicle communications at a specific road intersection. An adversary with extended scope can mount attacks against several parts of the

[1] Note that *stronger* refers here to an adversary with more capabilities such as wider coverage of the network, more receivers along the roads, more advanced data storage and processing power etc.

system in parallel and is not restricted to a specific region. An adversary with extended scope is also called a *global adversary*. In keeping with the layered meta model, scope can also be understood in terms of domains. An adversary with local scope attacks only one domain or parts of it, either vehicle domain, access domain, or backend domain; while an adversary with extended scope may attack two or all three domains at once. How scope is understood depends if the adversary model is domain specific or system wide.

The adversary model is now adapted to the three domains in terms of attacks against privacy. The results are summarized in Table 4.1.

In the *vehicle domain*, network membership does not provide advantages for the adversary. An insider would not be able to gain additional information from messages compared to an outsider, because safety messages would usually not be encrypted so that all traffic participants can benefit from it. Messages addressed to service providers may be encrypted, but network membership could not provide access to these messages. Furthermore, it is sufficient for an adversary to act passively in the vehicle domain, because active attacks would not yield more personal information than passive eavesdropping. An extended scope, however, provides an advantage over local attacks, because more data can be collected this way. More data, especially positioning information, facilitates tracking of vehicles for a longer time. Thus, we assume an adversary with extended scope.

For the *access domain*, membership makes a difference for the adversary. An insider may be able to control roadside units or intermediate communication nodes and collect data before it reaches the backend domain. For an outside adversary, data collection is more difficult if link encryption is utilized. An inside adversary is therefore a stronger assumption. The utilization of encryption also influences the method of an attack. In case of unencrypted communication between intermediate nodes, passive eavesdropping is sufficient to gather information. With encrypted communication an adversary has to actively gain access to the communication channel, either by taking control of a node, or imposing an intermediate node in a *man in the middle attack* [16]. Thus, active adversaries pose a higher risk. The scope of attacks against the access domain has implications similar to attacks against the vehicle domain. A successful local attack may provide an adversary with information that may be privacy sensitive. In a successful attack with extended scope more personal information can be gathered. Considering that we are assuming active attacks, attacks with extended scope in the access domain may be costly and may also be discovered more easily. Therefore, attacks with extended scope may be less likely than local attacks. We take the stronger assumption of extended attacks nevertheless. Note that attacks against the access domain can be mostly thwarted by utilizing end-to-end encryption between vehicles and servers (or service providers) of the backend domain.

The *backend domain* is where data is processed and probably stored for further usage. Typical components of the backend domain are traffic operation centers (TOC), databases, and service providers etc. An outside adversary may try to gain access to such a system by exploiting vulnerabilities. An insider may have to operate similarly, depending on the access control scheme in place. An insider may use additional knowledge which may facilitate an attack, thus an inside adversary is assumed. Attacks against backend nodes

have to be active, because it can be assumed that at least some security mechanisms are in place to protect client data. A successful local attack, i.e., an attack against a single backend node, already has a significant impact on the privacy of the original donors of the stolen personal information. However in an attack with extended scope, data stolen from several backend nodes could be combined and enable more detailed profiling. Therefore, we assume adversaries with extended scope.

	Vehicle domain	Access domain	Backend domain	Overall system
Membership	outsider	insider	insider	insider
Scope	extended	extended	extended	extended
Method	passive	active	active	active

Table 4.1: Summary of the adversary model for vehicle, access, and backend domain.

As is apparent from Table 4.1, the adversary models for the access and backend domain are stronger than the one for the vehicle domain. A generalized adversary model for the whole system has to reflect this. This implies that strong privacy protection must be provided in the vehicle domain already, and has to be extended throughout the access and backend domains. Therefore from an overall system perspective, an active, inside attacker with extended scope has to be assumed (Table 4.1, rightmost column).

4.4.3 Attack analysis

The information flow view and the adversary model are input for an analysis of potential privacy attacks. Therefore, we limit the attack analysis to potential information disclosure[2]. The attack analysis methodology employed by PRECIOSA is outlined below.

Attack trees [18, 19] are an established method for modeling security threats. They have already been successfully utilized for the modeling of attacks on inter vehicle communication systems [20], and will be used to model attacks on privacy for selected use cases in the following.

The main idea of attack tree analysis is to choose an aim or motivation of an adversary and to identify potential attacks that can be mounted to achieve that goal, and further assess the likelihood of these attacks. For that purpose, subgoals are defined for each attack that could lead to the specified goal. Subgoals are any means that are required to be successful with the attack on the level above, and can themselves be composed of more detailed subgoals. As a result, a tree structure evolves that defines a hierarchy of attack goals and subgoals. The root of the tree is the asset or attack aim, and nodes on lower hierarchy levels represent subgoals. The further down in the tree a node is, the more detailed and narrower is its description. Leaf nodes represent atomic subgoals that cannot (or do not require to) be further divided.

[2] General threat modeling approaches like STRIDE [17] consider a broader range of threats and potential attacks, not relevant to privacy. Note also that any non-policy-compliant access to personal data is here considered a information disclosure attack.

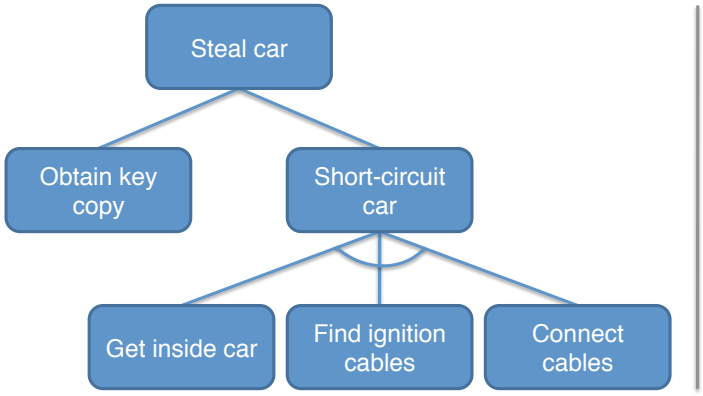

Figure 4.10: An example of a simple attack tree: tree representation (left side) and textual representation (right side) side by side.

Goals on the same hierarchy level either have an OR or an AND relation. In case of an OR relation, the nodes are independent attack alternatives that all can lead to the goal on the hierarchy level above. In case of an AND relation on the other hand, all subgoals have to be fulfilled in order to fulfill the goal above.

The concept of attack trees becomes clearer with a short example. We present the stealing a car example by Aijaz et al. [20]: The attack goal, and therefore the root of the attack tree, is *stealing a car*. To achieve this goal and drive away with someone else's car, an adversary can either *obtain a copy of the key* OR *short-circuit the car* – two alternatives that are both independently sufficient to meet the attack goal. *Short-circuiting* can be further split into a number of subgoals: an adversary has to *get inside the vehicle*, *find the ignition cables*, and *connect the cables* – all three subgoals have to be fulfilled together for the short-circuiting to be successful, they are part of an AND relation. Of course, *obtain a key copy* could be also further divided into subgoals, but we refrain from doing so for this short introductory example. Figure 4.10 depicts the resulting attack tree (left side). It is obvious that real-world and more detailed attack trees would become rather large, thus a textual representation can be used (right side of Figure 4.10) which is more compact than the tree representation. Then, subgoals are indented to reflect the hierarchical structure. OR and AND relations are specified for a goal and apply to the subgoals on the level below.

Attack trees can also be extended by assigning attributes to vertices, e.g., costs. Such information can be utilized to assess the likelihood or total costs of an attack branch in order to identify more serious attacks. For example, the seriousness of security threats could be classified using the DREAD methodology [21]. However, for our purposes of assessing privacy attacks cost attributes can be omitted. The intuitive attack tree approach is sufficient, because we are mainly interested in identifying privacy breaches without prioritizing between them.

Attack trees are used to analyze modeled architecture designs for potential privacy breaches. We utilize the textual representation rather than the graphical tree-like one in order to main-

tain clarity. Subsequently, the resulting attack trees are used to identify appropriate points of control and observation (PCOs) which can be used for measurement and evaluation to assess the privacy level provided by a given architecture. The details of this selection process will be discussed in the following section.

Example: Floating Car Data

In general, an adversary could either attack the identity privacy or the location privacy of a vehicle or its driver. While in the first case the aim of the attack is to obtain the real identity of a vehicle or driver, the recording of a moving pattern without learning the vehicle's identity may be sufficient for the latter. As a result, two attack trees have to be modeled for the FCD use case. One with the goal of determining the vehicle identity, and one aiming to track a vehicle. Note, that the tracking vehicle attack tree is also a sub-tree of the first one. The variables used correspond to the information flow view of the FCD use case (Fig. 4.9).

1. Goal: **Determine vehicle identity** V (OR)
   - 1.1. Link identifier I to V (OR)
     - 1.1.1. I is unique to V
     - 1.1.2. Anonymity set is too small (e.g., Swiss pseudonym in Sweden)
     - 1.1.3. Gain access to resolution authority that can map I to V
   - 1.2. Link position P to V (OR)
     - 1.2.1. P is specific to V (e.g., V is known to be there often)
     - 1.2.2. Link V to point of interest (POI) (e.g., home, or work) (AND)
       - 1.2.2.1. Track vehicle (see goal 2)
       - 1.2.2.2. Recognize POI specific to V
   - 1.3. Infer V from traffic status data T (e.g., T contains quasi-identifiers of V)
   - 1.4. Link destination D to V (e.g., D is specific to V, only V ever goes to D)
   - 1.5. Infer V from routing information R (e.g., R contains quasi-identifiers of V)
2. Goal: **Track vehicle** (OR)
   - 2.1. Link messages with spatio-temporal correlations (e.g., multi-hypothesis tracking)
   - 2.2. Link messages with different identifiers (e.g., observe identifier changes)
   - 2.3. Link messages with different traffic data (e.g., similar message content)

The attack tree shows that with the exception of corrupting a resolution authority (1.1.3), which requires active engagement, an active adversary does not gain any advantage over an adversary passively eavesdropping on communication. Subgoals for determining a vehicle's identity (goal 1) only require local scope, except for 1.1.2. Tracking a vehicle on the other hand requires an adversary with extended scope to be able to successfully link position samples or identifiers over time. An adversary does not have to be an insider for both attacks to succeed, and inside knowledge may only be helpful for gaining access to a resolution authority's identity mappings. However, gaining this knowledge would require strong determination and resources to become a legitimate insider of such an authority without being identified as an adversary.

4.4.4 Identification of PCOs

Based on the modeled information flow and the attack analysis, we can now determine places in the system which are suitable to measure and verify the privacy of the modeled architecture. These points are called points of control and observation (PCOs).

In a first step, data items containing personal information are identified from the results of the attack analysis and privacy goals are formulated for these items.

Subsequently, the identified attacks and privacy goals are matched to the information flow view. Whenever privacy goals collide with potential attacks as well as at identified high risk attack points a point of control and observation (PCO) should be set to be able to assess and evaluate the provided level of privacy at these points.

As already outlined in Section 3.4, some generalized PCOs can be identified that are applicable to arbitrary architectures. The level of privacy should always be measured when personal information leaves the entity it originated from, e.g., the vehicle. Furthermore, PCOs should be set at domain borders to ensure that privacy is not subdued in the next domain. Most importantly, privacy has to be assessed with PCOs whenever information is being modified, processed, stored, or combined with other information.

Example: Floating Car Data

The attack analysis performed in the previous Section provides valuable information about which data items pose potential privacy risks in the FCD use case. Using this information we can formulate the following privacy goals for this use case:

- The identifier I should not be linkable to the vehicle V or other identifiers used by V at a previous or later point in time.
- Position information P should not be linkable to vehicle V or other position samples corresponding to V.
- Traffic probe information T should not contain information linkable to V.
- The destination information D should not be linkable to vehicle V.

- Routing-relevant information R from the TOC should not be linkable to identities of data subjects.

The first three points are concerned with the privacy of data items originating from the vehicle (I, P, T) when reporting floating car data to the TOC. Therefore, it is necessary to verify the privacy of these data items when they leave the vehicle. The items should be considered together for measurement and evaluation because the relationships between the items could also help an adversary to identify a vehicle and thus likely also a data subject.

In inter-vehicle communication (I,P,T) may be modified in some way or cached before the data is forwarded to a TOC. For example, vehicles receiving the data may aggregate it with their own sensor values, thus changing I (e.g., replace it with own identifier, or add own identifier to a list of identifiers), modifying P (e.g., an extended region instead of a single position sample), and T (e.g., supporting T or averaging with own values). Thus, privacy has to be also measured whenever the data passes through another vehicle, regardless of the actual operations on the data.

The access domain provides a channel controlled by the network provider. It is essential to measure and verify the privacy of data that enters the access domain to guarantee that caching of that data does not pose additional privacy risks. Privacy has to be verified at each intermediate node from vehicle to the backend domain. In some cases other vehicles may forward messages as well. Because different additional information may be available in vehicle or access domain, two separate PCOs are recommended to properly measure the privacy level at forwarding nodes.

When data is received in the backend domain, privacy has to be verified when data is stored in the TOC's database, because storing it together with other traffic probe data may enable inferences of quasi-identifiers. Privacy should also be measured when stored probe data is processed and fused with other data to model the real-world traffic situation. The resulting model must maintain privacy.

The fourth and fifth privacy goals, specified above, apply to the logical function of requesting and consuming up-to-date traffic information from the TOC. For the request message (I,P,D) the same PCOs apply as for reporting traffic status data.

In the backend domain, the TOC processes the request results using a relevant subset of information on the current road and traffic situation relevant for route calculation from the vehicle's current position P to destination D. The processing (W) of the request (I,P,D) together with stored traffic probes may enable privacy breaching inferences and therefore has to be measured. For the resulting information R the privacy with respect to the original data subjects has to be measured before it is send back through the access domain to the requester.

Table 4.2 provides a summary of the recommended PCOs for the floating vehicle information use case together with information about where they have to be integrated and what data has to be measured. Figure 4.11 integrates the PCOs with the information flow model of Figure 4.9.

PCO	Domain	Integration point	Critical data items
1	Vehicle	prior to sending	(I,P,T), (I,P,D)
2	Vehicle	message forwarding	(I,P,T), (I,P,D)
3	Access	message forwarding	(I,P,T), (I,P,D)
4	Backend	prior to storage or processing	(I,P,T), (I,P,D)
5	Backend	on data fusion	(I,P,T)
6	Backend	after processing	R

Table 4.2: Recommended PCOs and their integration for the floating vehicle information use case.

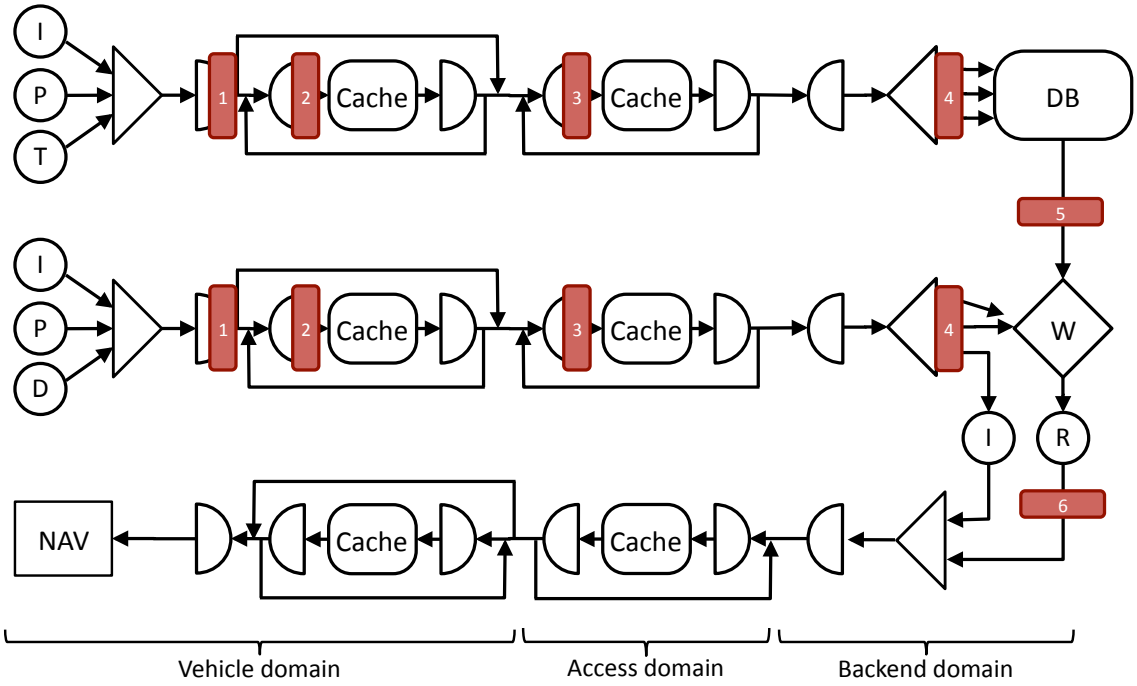

Figure 4.11: Recommended points of control and observation (PCO) for the FCD use case. The information flow model of Figure 4.9 is overlaid with PCOs (red, Table 4.2) that are suitable to measure, assess, and verify the provided level of privacy.

4.4.5 Evaluation of PCOs

Having identified all PCOs representing points where a privacy breach is possible, we will now evaluate the system design using several metrics. The choice for a given metric may depend on the given application as well as the statistical background information available for verification and can be different for each identified PCO. We will now evaluate the PCOs that have been identified in the FCD example application in the previous sections

using the metrics defined in deliverable D2 [12]. In summary the following steps will be taken to judge a system:

1. We can observe the number of PCOs identified in a given system. This number will give us a first rough estimate of how privacy preserving the system is. The more occasions there are where a certain privacy level has to be ensured by taking measurements, the less probable it is that a given system is privacy friendly.
2. For each PCO, the involved data can be traced back to the identified attack goals using the attack trees defined in Section 4.4.3. This gives us a sense of which involved data items are sensitive. Thus we call this approach the *tainting of sensitive data*. Having identified sensitive data, the attacker model can then be used to decide whether an attack is feasible at a certain PCO. This gives an overview of privacy issues in the system design and unveils at which points more fine-grained analysis is required. The tainting metric complements the approach presented in deliverable D2 Chapter 6 [12] from a PCO centric point of view.
3. Depending on the results of applying the first two very general metrics to a given application, more sophisticated metrics can be looked at. Candidates here are the concept of k-anonymity and entropy based measures. It needs to be decided per application which metrics are applicable at each PCO; deliverable D2 [12] gives several guidelines on how to select appropriate metrics.

All steps will be further discussed and exemplified using the Floating Car Data use case that has already been used throughout this chapter.

Example: Floating Car Data

Number of PCOs As detailed in Table 4.2, a total of 6 PCOs has been identified. Because information travels through vehicle, access, and backend domain mostly unencrypted, a potential attacker can extract information at all domain borders and several points of data aggregation.

Tainting of Sensitive Data To get a general idea of the flow of private data, we will next measure how personal or potentially linkable data travels the network. That is, we trace each goal of a potential adversary back to the required information using the attack trees that were modeled by following the tree's edges down to the required data items. We can then use this knowledge to look at each PCO in turn to see which potentially personal information is involved and to what entities it is sent. Comparing this to the abilities of a possible adversary as defined by the adversary model (cf. Section 4.4.2) will give a sense of which attacks are feasible at certain points in the system. In deliverables D2 (Chapter 6) [12] and D6 [22], a similar metric is presented to measure system privacy as a whole. At this point however we are concerned with whether a given attack is feasible at a certain point of the system. The higher the uncertainty using this general metric is whether an attack is feasible, the more likely we need to employ a more sophisticated metric.

PCO 1 The data items involved at PCO 1 are an identifier of some kind (I) – not to be confused with a vehicle identity (V) –, a vehicle's current position (P), a traffic status report (T) and a destination (D). Assuming that T does not contain any quasi-identifiers of V, the traffic status report T will be of little to no interest for an adversary. All of I, P, T and D however can be traced back to a potential goal given the modeled attack tree.
PCO 1 is located in the vehicle domain. The information flow view shows that data is not encrypted at this point in time and that its destination is other vehicles – or caches – in the vehicle domain. Because of the missing encryption, an adversary without insider privileges can freely access and read the data. Assuming an adversary with extended scope as detailed in Table 4.1 he will be able to collect several of the said data items in a larger area and over an extended time.
We can therefore deduct, that goal 2 – the tracking of a vehicle – is feasible for an adversary given the assumed membership, method and scope. Goal 1, determining a vehicle identity (V), can in general be assumed to be possible considering the type of data going through PCO 1 and the assumed scope of an adversary. For example an attacker can learn typical destinations of vehicles by observing D. But detailed elaboration requires more complex metrics and further knowledge about the specific application and the setting in which it runs, like the scope of deployment (i.e., state-wide, or country-wide) or the number of users.

PCO 2 At PCO 2, the same data items as for PCO 1 are involved and it is located in the vehicle domain as well. The way the given example system was modeled, there is no data fusion in the vehicle domain. That means that forwarded information passing PCO 2 is an unaltered version of the items passing PCO 1. The more forwarding steps are present however, the more potentially personal information is spread throughout the network.
As for the feasibility of attacks, the same arguments as for PCO 1 hold, augmented by the aforementioned further spreading of information which makes successful tracking more probable.

PCO 3 Here, again, the same data as for PCO 1 and 2 is involved. PCO 3 is located in the access domain however. In the access domain we assume an attacker that has insider privileges, again extended scope and is able to perform active attacks. Thus, since no information alteration or fusion has happened so far, we have to assume that it is feasible for an attacker to gain knowledge about several location samples of an identifier thus making tracking possible. This is especially bad at the access domain because location information of several vehicles is decoupled from the physical location of the vehicles. That is, an attacker does not need to be physically present in the vicinity of the tracked vehicles but can intercept tracking data at other points in the access domain which might be more convenient to attack. All other possible attacker goals as well remain generally feasible. Moreover, an active attacker might even inject false or specially crafted data in the forwarding process to evoke the sending of more possible sensitive data of a vehicle.

PCO 4 By now the aforementioned information – still in its unaltered form has entered the backend domain. Because of the same assumed attacker model in the access and

backend domain, the same argument holds as for PCO 3. But, again, information is even more decoupled from the physical entities it is about. Thus, it is even easier for an attacker to collect data at this PCO.

PCO 5 This PCO is located at the first point where collected data is fused to gain a new, higher level information. The data items used here are I, P and T. It can be assumed that many information triples of this kind travel through this PCO to be fused into traffic information about a certain road segment in the next step. That of course means that an attacker at this point gains knowledge about already complete tracks of several vehicles. Goal 2 therefore remains feasible. The same argument can again be made for Goal 1 because of the amount of data involved at this PCO. If however the database is queried only for averaged data to be applied to the traffic status calculation, this is the first PCO for which tracking of singular vehicles will become infeasible for an adversary.

PCO 6 Now, after data fusion, a new information item, namely the route information R has been generated. An attacker observing R at PCO 6 with the adversary model assumed in the backend will not be able to infer information about the vehicles contributing information to R anymore given that enough vehicles have contributed or, more precisely, that enough information has been passing PCO 5 recently. This already implies that those two PCOs again need to be further discussed using more complex metrics. Apart from the traffic information contained in R, an attacker might infer information about the possible destination of a driver. This information is however not directly contained in the information passing PCO 6 anymore.

k-Anonymity Measurement

The previous discussion showed that the nature of the data at several PCOs is very similar. Thus, the discussion for the application of k-anonymity will be grouped in the following. For general guidelines on how to apply k-anonymity based measurement to PCOs we refer the reader to deliverable D2 [12].

PCO 1–3 As outlined in the previous Section, the data items I, P, T and D can be observed at these PCOs. For the measurement of k-anonymity we either need current traffic status information or statistical data that gives us the minimum amount of vehicles passing a certain region in a specific time interval. Given that data we can measure the k-anonymity regarding the observed time and position tuples of vehicles based on the granularity of information provided. Suppose for example that in case of the FCD application, a minimum of 20 vehicles pass any given 1000 m road segment in a one minute period. If vehicles only send out their current position with a precision of 1000 meters, we can then deduct that $k = 20$. While this adaption of location granularity might be acceptable for the FCD application, it might be far to coarse-grained for safety applications. Another aspect is that the assumption of a certain minimum amount of vehicles passing at a given time interval might be hard to make because even roads with known high vehicle density might drop during a certain threshold over night.

If no assumptions can be made about minimum vehicle density or an acceptable amount of artificially coarse-graining location information, we have to assume $k = 1$ regarding location privacy given that no information fusion or obfuscation happens in the example FCD information flow at any of the PCOs 1–3.

PCO 4–5 In general, the same argument regarding k-anonymity holds as for the PCOs 1–3. However, being located in the backend domain, PCO 4 offers an easier access to larger amounts of data at the same time. That is, we do not need to make observations for a longer time to deduct statements about k-anonymity here. Note that, as already hinted at in the last section, only average queries are allowed to the database, k-anonymity will rise dramatically at PCO 5 because an adversary will only get information about longer tracks that involves indistinguishable information about several vehicles.

PCO 6 The only data item observed at PCO 6 is the calculated route information R. Because of the data fusion that has happened to generate the route information, the anonymity set is likely to be high for the vehicles that donated the traffic status information which served as input to the route calculation. In general one can say that once data has been averaged or fused in some other way which makes the single data items that went into the calculation indistinguishable, the size of the anonymity set k will be as high as the number of data items that were fused.

Entropy-based Measures

Similar to the calculation of k-anonymity we will group the discussion of entropy based measurement in the following according to the data items that can be observed at the PCOs.

PCO 1–3 Again, the data items I, P, T and D can be observed at these PCOs. The following will outline how each of them can be used to serve as input to an entropy based metric presented in deliverable D2 [12] that gives the entropy in a system based on the certainty about a mapping from an individual to an origin to a destination and back to the individual again. If an adversary can observe a pair of P and D he can deduct with high probability that P is the origin of a certain trip and D will be the destination. It then only depends on the implementation of I (e.g., pseudonym or unique identifier) and on other knowledge of the adversary about P or D which might be known to belong to a certain individual how much entropy is still left in the system.
If an adversary does not observe D but only several pairs of time and location information, this too can be used to track individuals and thus reduce the uncertainty about how origins map to destinations.

PCO 4 As for the k-anonymity measurement, an adversary needs to observes several messages over time to deduct useful information for entropy based measurement. This amount of time can be reduced if an adversary observes messages at this PCO because more data is available in a shorter time period.

PCO	Tainting	k-anonymity	Entropy
1	privacy relevant data freely available	likely to be small	likely to be low due to (P, D) pairs
2	privacy relevant data freely available	likely to be small	likely to be low due to (P, D) pairs
3	privacy relevant data freely available	likely to be small	likely to be low due to (P, D) pairs
4	privacy relevant data freely available	likely to be small	likely to be low due to (P, D) pairs
5	privacy relevant data hidden if database only allows aggregated queries	high if database only allows aggregated queries	might reduce k-anon. if earlier PCOs observed as well
6	only indirect deduction of privacy relevant data possible	likely to be high	might reduce k-anon. if earlier PCOs observed as well

Figure 4.12: Privacy level of the example FCD system.

PCO 5–6 Here, the combination of observations at different PCOs can lead to reducing the entropy of a k-anonymity measurement. As discussed before, it is possible that at PCO 5 only already averaged or otherwise fused data can be observed. If not at PCO 5 due to database implementation, this is definitely the case at PCO 6. It was argued that in this case the size of the anonymity set k will be equal to the number of data items that were fused at this point. However, if an adversary has observed further information at other PCOs, this might lead to a lower entropy value than the value of k implies. Suppose for example that by observing data at earlier PCOs an adversary already knows that certain vehicles contributed to a fused value. Then the overall entropy is less than the theoretical maximum of $\log k$. Thus the anonymity of the other vehicles also contributing to the fused value whose data has not been observed before is also reduced.

The previous elaborations have shown that first of all, there are many points of control and observation in the example information flow for the FCD application. Many of those make it feasible for an adversary to breach privacy of the system's users. Table 4.12 gives an overview of the PCO evaluation. The main problem of the modeled system is that no encryption is in place at any point so that data is freely accessible by many possible adversaries in the vehicle, access and backend domain. An alternative to encryption of data would be fusion (e.g., averaging) of data over certain location intervals as early as possible thus raising k-anonymity values at later PCOs. In conclusion this analysis already shows at design time that the resulting system would likely have a low privacy value during runtime.

Note that while it is not possible to do exact measurements at design time of the system, this analysis can serve as input for more precise measurements during runtime. Furthermore, during a runtime analysis, the system can dynamically adapt to ensure con-

formance to k-anonymity thresholds or other measures by making forwarded information more coarse-grained.

4.4.6 Comparison of Alternative System Designs

Sections 4.4.1 to 4.4.5 already outlined how the privacy level of a given system design can be evaluated based on information flows. As a result of the last step, the evaluation of PCOs, one gets a good idea of a system's privacy value. A system that provides no or only weak privacy will have a low value, while systems providing strong privacy will have a corresponding high privacy value. Privacy values at the design phase of a system are mostly determined by the number of PCOs, the estimated k-anonymity values as well as the estimated entropy values.

This way, different system or architecture designs for the same application can be modeled independently, but their privacy levels can be easily compared with information flow analysis. The information flow of each variant is analyzed accordingly and resulting privacy values help to assess which solution is the *best* in terms of privacy for the user.

However, a system design may be privacy-friendly in most parts but provide only weak privacy protection at certain subsystems. Rigorous information flow modeling with the previously outlined approach should detect such weaknesses by placing PCOs at critical components. While a resulting low privacy value alone would not be useful for a system designer, in combination with the detailed information flow analysis and the PCOs as verification points the privacy flaws in a given design can be pinpointed quite accurately. This enables a system designer to revise a given design to enhance its privacy properties, e.g., considering a different deployment strategy of logical functions. The new design can then be modeled in the layered reference model and information flow analysis can be applied. By comparing the privacy level of the new design with the privacy level of the old one it is immediately apparent which solution provides better privacy.

So information flow analysis can serve two purposes: it can be employed to compare the privacy levels of independent system designs for the same application, or the privacy of one system design can be iteratively enhanced by identifying weaknesses, revising the design, and evaluating the result again with information flow analysis. How information flow analysis is used, depends strongly on the development methodology that is employed.

For the analysis of several system designs parts of the information flow analysis of the first variant may be reused. The information flow of each system design variant can be modeled on its own or adapted from previous designs. PCOs have to be identified and evaluated separately for each variant, as different designs might have different ways of attacks. However, it may be possible to reuse previous work from one variant only adapt them to the new variant. Attack trees have actually been developed with reusability in mind, either of the whole tree or certain subtrees. The adversary model on the other hand, should always be reused in order to maintain comparability between different analyses. Assuming a different adversary would prevent straightforward comparison between analyses.

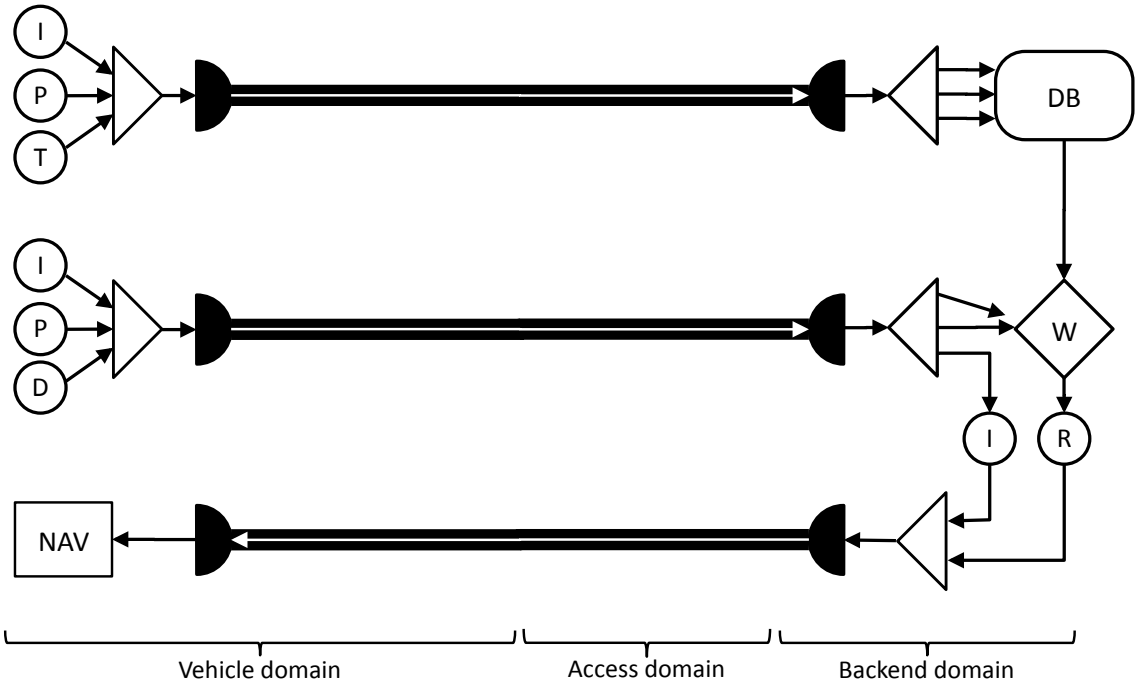

Figure 4.13: Information flow view of an alternative system design for the floating car data use case with a secure channel between vehicle and server.

Example: Floating Car Data

In the previous Sections, floating car data served as a continuous example for information flow analysis. The given system design for FCD has been fully modeled and analyzed. In this Section, we introduce an alternative system design for the FCD use case that employs encryption to establish a secure channel between vehicle and TOC server. We shortly discuss the information flow analysis of the alternate version and then compare the privacy of both variants in order to determine which solution is more privacy friendly. In the following, we will refer to the FCD system design discussed in the previous Sections as *basic FCD* and to the alternate version with encryption as *enhanced FCD*.

Information flow of enhanced FCD. In the enhanced FCD use case, the same information items are created at the vehicle but a secure channel to the server is established before traffic data is sent, e.g., encrypting the message asymmetrically with the well-known public key of the TOC. Similarly, a secure channel is established before a vehicle sends a route request, and the server would also use a secure channel to the vehicle to return routing-relevant information. Figure 4.13 depicts the information flow for enhanced FCD.

Attack analysis of enhanced FCD. The same adversary model is used for basic FCD and enhanced FCD. Because the information items remain the same for enhanced FCD, the attack tree for basic FCD can be reused as well. However, due to the utilization of secure channels between vehicle and server the number of potential attack points is reduced. See Section 4.4.3 for the corresponding attack tree.

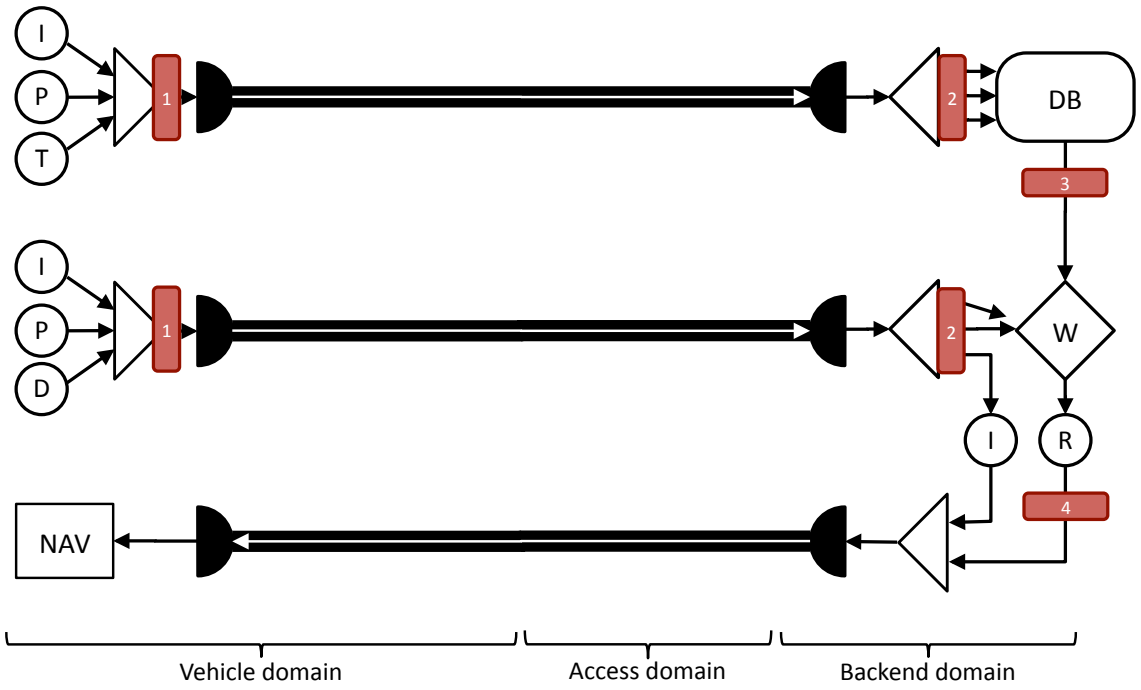

Figure 4.14: Information flow view of enhanced FCD use case with corresponding PCOs.

Identification of PCOs of enhanced FCD. The identification of PCOs is analogous to the basic FCD use case. Privacy-relevant information items remain the same and, therefore, privacy goals do too. Due to the secure channel between vehicle and server less PCOs are required. Forwarding nodes in vehicle and access domain can be ignored because transferred information is opaque to them. Figure 4.14 depicts the PCOs for the enhanced FCD, determined based on the process outlined in Section 4.4.4. Before information leaves the vehicle privacy has to be measured. In transit, data is encrypted and privacy verification is not required until data is received by the server and inserted into the database. PCOs 3 and 4 correspond to PCOs 5 and 6 of the basic FCD. Table 4.3 summarizes the PCOs for the enhanced FCD use case.

Evaluation of PCOs of enhanced FCD. The identified PCOs for the enhanced FCD use case are a subset of the PCOs of the basic FCD. Table 4.4 shows the mapping of the PCOs to each other. The evaluation of the four identified PCOs is analogous to the

PCO	Domain	Integration point	Critical data items
1	Vehicle	prior to sending	(I,P,T), (I,P,D)
2	Backend	prior to storage or processing	(I,P,T), (I,P,D)
3	Backend	on data fusion	(I,P,T)
4	Backend	after processing	R

Table 4.3: Recommended PCOs and their integration for the enhanced FCD use case.

evaluation of the corresponding PCOs of the basic FCD discussed in Section 4.4.5. Note, that no PCOs are required in the access domain, reducing the total number of PCOs from 6 to 4.

basic FCD	PCO1	PCO2	PCO3	PCO4	PCO5	PCO6
enhanced FCD	PCO1	-	-	PCO2	PCO3	PCO4

Table 4.4: Relationships between PCOs of enhanced FCD and basic FCD.

Comparison of results. When comparing basic and enhanced FCD designs, it becomes apparent that the enhanced FCD system design provides better privacy than the basic FCD design. The enhanced FCD requires less PCOs, completely eliminating the necessity to verify privacy in the access domain. While in the basic FCD design all intermediate forwarding nodes had access to the information, only vehicle and destination server have access to the information in the enhanced version. For the remaining four PCOs privacy evaluation is equivalent to their counterparts in the basic FCD design, thus the enhanced design does not provide any further privacy enhancements here. Nevertheless, by reducing the number of entities that handle privacy information to the information origin and the information sink the chance that privacy-relevant information will be exposed is reduced.

Further optimizations of the system would be possible by introducing data fusion in the vehicle domain eliminating the need to send exact information to the backend domain. However, the following Chapter will outline how mandatory privacy control and integrity protection of the mandatory privacy control in the backend system will allow to ensure that no privacy infringements are possible for operators of the backend system. Thereby the introduction of end-to-end encryption from vehicle to backend domain in combination with runtime enforcement of privacy protection at the backend domain can be enough to achieve a high privacy level of the system.

5 cITS Runtime Architecture - Architecture Framework

5.1 Overview

Chapters 3 and 4 are concerned with the design of a privacy-friendly ITS architecture and introduced the basic concepts of a design methodology that allows creation of such architectures and respective applications. In this section, we will introduce those parts of our cITS architecture framework that concern runtime components.

Chapter 2 defined hippocratic principles for privacy-aware cITS, or HcITS. From these principles some major conclusions were drawn on how these principles can be integrated and reflected in a cITS architecture. In the following, we refine these general concepts and give an overview of components required to implement our privacy principles and, thus, achieve the following goals:

From faith to guarantee: In today's systems, users normally do not get technical privacy guarantees. When they provide personal information or person-related data, e.g., on a web site or to a navigation provider, the provider of this service might – in the best case – publish a privacy policy that declares how that provider intends to use that data. More often than not, even this is missing and the user can only hope that the provider respects national privacy laws. His only other option is not to use the service. We intend to reverse this situation: the user should specify policies, give the data to the provider, and the system should guarantee that these policies will be respected.

System privacy: The PRECIOSA runtime architecture covers the whole cITS, i.e., the vehicle domain, the access domain, and the backend domain. It is not limited to any single area and will be flexible enough to also cope with future application demands.

An architecture that realizes these goals will be able to enforce privacy protection in the complete system at runtime. We therefore term it the *Privacy-enforcing Runtime Architecture* or *PeRA*. We will now first outline fundamental concepts we derived from the hippocratic principles and requirements on which our runtime architecture is based. Afterwards, major architecture building blocks are outlined which extend the generic cITS architecture of Chapter 2 and transform it into a truly hippocratic cITS architecture.

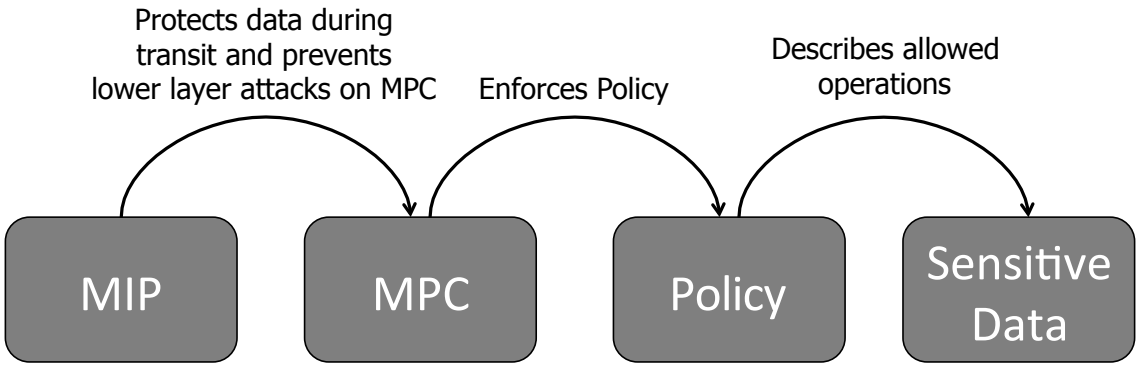

Figure 5.1: PeRA Protection Mechanisms that define the policy enforcement perimeter.

5.2 Policy Enforcement Perimeter

In Chapter 2, we outlined a generic cITS system architecture as it can be found in many ITS systems. However, this generic architecture lacks inherent support for privacy preserving mechanisms. At the same time, hippocratic principles and privacy requirements have been defined. In order to realize these principles and requirements we first have to establish trust in other nodes by verifying that they implement and follow these principles. It needs to be guaranteed that users can set policies for their personal information and that these policies cannot be broken by any data processing entity. For this, we need to establish a trust domain that extends beyond local system borders. We call this trust domain the *policy enforcement perimeter (PEP)*. Inside this perimeter, data is guaranteed to be always coupled with according metadata, i.e., policies. Further, adherence to the policies contained in the metadata is mandated throughout the PEP. This is achieved by two fundamental concepts, derived from the hippocratic principles and requirements.

The first function is termed *Mandatory Privacy Control* (MPC) which is a reference to the Mandatory Access Control (MAC) schemes found in access control architectures like Bell-LaPadula [23]. Just like it is not at the user's discretion to change access policies in mandatory *access* control schemes, it is not at the application's discretion to change privacy policies in our mandatory *privacy* control system. To the contrary, it is mandatory that applications have to obey the user-defined privacy policies whenever personal information is accessed. The second function is termed *MPC Integrity Protection* (MIP). The goal of MIP is to establish trust in remote systems and to ensure that only qualified applications are able to access and process sensitive data.

If the application or the MPC mechanism is tampered with, the MIP will detect the integrity violation and will prevent further data access. Starting from the personal data and the related policy, the functions form a chain of control instances, as shown in Figure 5.1, that defines the Policy Enforcement Perimeter, in which non-policy-compliant data access is prevented.

5.2.1 Mandatory Privacy Control

We assume that access to personal information is exclusively done via a query-based API. It is the responsibility of the MPC to ensure that only those queries are executed that are in compliance with the accessed data. The query interface is accessed by so called *uncontrolled applications*, that is, applications running outside the PEP. Data that is returned as results to such queries is guaranteed to be policy compliant, however, it will not be further protected by the PeRA. Therefore, an unrestricted application is able to process and share the query results in any way it wants. Nevertheless, query results are still accompanied by a privacy policy. While enforcement of the policy cannot be guaranteed, some applications may still choose to comply. The result is best effort policy compliance outside the PeRA protection perimeter. If an application needs broader access to sensitive data, the concept of *controlled applications* can be used (see Section 5.3.4).

To guarantee that the MPC is implemented correctly at each cITS node, a (potentially manual) certification process is necessary. Vehicles will encrypt and transmit their sensitive data only to other systems that are certified accordingly. We argue that the verification process is performed manually, as automatic verification of correct implementation of MPC is extremely hard to achieve as it would require an automated correctness proof of corresponding architecture components. Therefore we envision that a trusted third party will examine the MPC implementation and integration in the overall system. The correct verification will then be expressed by a cryptographic certificate issued for the public keys of this system. Note, that this manual certification process only applies to the generic PeRA core and is not needed for specific applications. All access to sensitive data by the PeRA itself will then be bound to the exact functional state in which the architecture components have been certified.

Note further, we assume that we have to guarantee the compliance of the system regarding any given set of policies. At this point we do not consider the correct specification of such a set of policies for a given application. That would be another verification process for the application at design time.

5.2.2 MPC Integrity Protection

The concept of MIP requires that all personal information is kept confidential for instance by using encryption, when it leaves the data subject's immediate control. We assume that this is the case as soon as it leaves a cITS node, e.g., a vehicle. If data would be sent unencrypted, any receiver of this data would be able to use it for arbitrary purposes and the compliance to a privacy policy could not be guaranteed any more.

Besides confidential unicast communication, some applications may also require other types of trusted communication, e.g., in a certain group of vehicles. If for reasons of application characteristics, data needs to be sent in an unencrypted way, it has to be anonymized or otherwise altered beforehand so that it does not constitute personal information anymore.

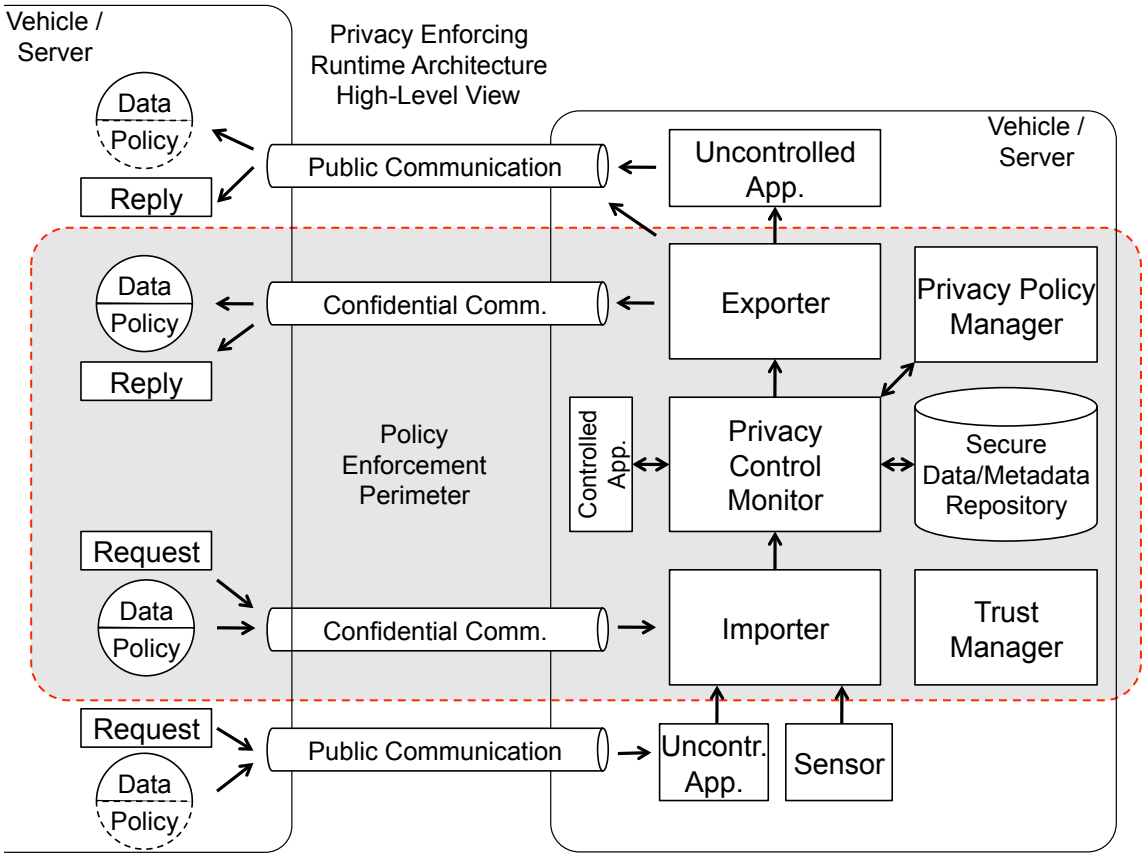

Figure 5.2: High level view of PeRA and its architecture building blocks.

Data encryption must be realized in a way that only cITS nodes that implement the MPC and therefore guarantee to comply with the privacy policy can decrypt and work with the data. In other words, MIP must guarantee the integrity of system components that are able to decrypt the data and provide policy-compliant access for applications. If there is any modification or tampering of the MPC component, decryption and access to the data must be refused by MIP.

5.3 Architecture Building Blocks

So far, we have defined the fundamental concepts necessary for establishing a Policy Enforcement Perimeter. Now, we extend the generic cITS architecture from Chapter 2 accordingly. Figure 5.2 shows the resulting HcITS architecture. The PeRA introduces several new components and major building blocks to realize the hippocratic principles. Note that all interacting cITS nodes should run the PeRA. As already discussed in Chapter 2, personal information should always be combined with a privacy policy that governs how this data can be used. It is the objective of PeRA that this policy can never be violated.

The policy enforcement perimeter requires confidential communication between different nodes for data exchange as well as request-reply style communication. But public communication needs to be considered, too. Application-specific requirements or communication scenarios may prevent the use of confidential communication, e.g., in broadcast situations. In such a case, orthogonal mechanisms are required to achieve privacy. Section 5.3.5 elaborates further on public communication.

Furthermore, importer and exporter components have been introduced. The importer has the purpose of treating all data that enters a local system equally. This applies to local sensor data, data-metadata tuples received from other cITS nodes inside the Policy Enforcement Perimeter, as well as public data that may or may not be accompanied by a privacy policy. The importer ensures that a policy is assigned to any data item before it is processed or stored in the local secure data/metadata repository. For example, default policies are assigned to local sensor values, depending on the sensor type. The exporter provides a unified output interface that enables confidential communication with other HcITS nodes inside the Policy Enforcement Perimeter, but also provides public communication support for applications that have to rely on broadcast. It further provides the query interface for uncontrolled applications that need access to data. This query-based API can either be generic (e.g., SQL) or it can be application specific (e.g., tailored for accessing traffic information).

5.3.1 Privacy Control Monitor

The privacy control monitor (PCM) is the core building block of the MPC implementation. The PCM has to fulfill several functions. First and foremost, the PCM acts as a privacy middleware between personal information of data subjects and applications that want to access the data.

The PCM evaluates polices of requested data items and has to decide whether operations are permitted, rejected, or only permitted under certain anonymization restrictions. Applications that want to access data need to specify a purpose, and the PCM matches this purpose and the role of the requesting application with the policies of the requested data items. This way, the principles of purpose specification and limited use are realized.

Further, when queries are executed, it is possible that policies of different data subjects must be merged, possibly resulting in conflicts. This is detected by the PCM. Resolving of conflicts and merging of policies is, however, left to the privacy policy manager (see. Section 5.3.2). The PCM then enforces the resulting policies given by the privacy policy manager.

In order to detect privacy or security breaches, a privacy trail should be kept to analyze the sequence of events such as applications submitting a query or sending a message to another HcITS node. What information needs to be collected is, however, up for a specific design. Furthermore, requests and queries should be analyzed for patterns that might suggest unauthorized data collection to detect privacy intrusions. For example, a series of single queries could be used to aggregate data that, according to its policy, is not to be combined with other data.

In accordance with the principle of limited retention, privacy maintenance functionality is also required. Data should be deleted once the allowed usage period specified by the corresponding policy is up. We realize that it might be necessary to make data inaccessible (i.e., *logically* deleting it) rather than *physically* deleting it due to governmental or legal requirements. How to manage this trade-off and how to "seal" data such that it becomes inaccessible under regular terms needs to be addressed by a specific architecture.

5.3.2 Privacy Policy Manager

The Privacy Policy Manager is responsible for the relationship between data and policies. It also translates user-specified privacy preferences into default privacy policies. Individual policies that have been specified by the data subject are the input for the Privacy Policy Manager that generates the necessary structures to store this metadata in a machine accessible form. Further, the Privacy Policy Manager offers the functionality to merge policies and resolve arising conflicts. For example, the policy specified by one data subject may define different requirements than that of another data subject. Also, different stakeholders, e.g., data providers or national laws and regulations, can pose restrictions on the data that conflict with the data subject's settings. Those conflicts must be detected and possibly be reported back to the data subject.

The Privacy Policy Manager cooperates closely with the PCM. Basically, the Privacy Policy Manager handles storing, fetching, and merging of policies, as well as conflict resolving for the PCM while the PCM performs policy analysis to infer access and data processing decisions.

5.3.3 Trust Manager

The Trust Manager is the core building block of MPC Integrity Protection. Its responsibility is to measure the integrity of the other PeRA components and to determine whether the system is in a trusted, certified state. Therefore, the Trust Manager acts as a local trust anchor for remote HcITS nodes. How the Trust Manager is implemented depends on the platform and type of the cITS node. For example, a hardware security module can be used for trust establishment. Trust mechanisms will be discussed further in Chapter 6 and deliverable D10.

5.3.4 Controlled Applications

As indicated earlier, there will be applications or services that need to directly access and process personal information. An example is a request to a traffic information service where a vehicle wants to check the traffic situation on its planned route. While these calculations could also be performed locally by the vehicle, shifting them to the backend may be desirable out of different reasons, e.g., bandwidth considerations or data not available at the vehicle. The vehicle will have to send its position and route information to a traffic

information service. Giving this data to an unrestricted application would jeopardize our goal of privacy protection enforcement, as we cannot guarantee that the data is used in a policy compliant way (e.g., only for this single request and only for retrieving suitable traffic information).

We therefore introduce the concept of *controlled applications*. A Restricted Application can run inside the Policy Enforcement Perimeter, but is restricted in its capabilities. For this purpose, controlled applications are executed inside a controlled environment, e.g., a sandbox environment, that limits communication with external components. For such applications persistent storage is only possible in the secure data/metadata repository via the PCM, which will check compliance with privacy policies.

What kind of restrictions apply to these applications depends on their service. For request-reply interaction, replies could be restricted to the original requester. It also possible that controlled applications are completely transient, i.e., they cannot store any internal state between requests. All state is deleted by the runtime environment between invocations. It is important to limit their functionality according to the principles of limited collection, limited use, and limited disclosure.

To achieve the latter, controlled applications that need to communicate with external services can only do so by means of a privacy proxy. The privacy proxy will check the compliance of communication with the privacy policies of involved data. One possible way to achieve this is the analysis of the outgoing data by a component that knows the semantics of the data exchange format. However, even if the exchange format is strictly specified, covert channels could still exist and allow an adversary to leak private information without the user's consent. An alternative is to analyze the restricted application itself either by means of static code analysis similar to the approach presented in [24] or by means of a special script language for controlled applications, comparable to that presented in [25], that annotates private information and can be verified on the fly at runtime.

However, whether this analysis is needed, depends on whether the restricted application needs external services. If a restricted application can work locally on the server only with the data stored there, which is accessed using the query based API, then no further analysis of the application code is needed. The controlled application environment itself will then prevent policy violations. This means that different classes of controlled applications are possible: The more external communication is needed, the stronger the requirements on the verification of the application code for policy compliance are.

Controlled applications will on the one hand be able to process sensitive data given to them in service requests by the data subjects. On the other hand, strict policies can prevent storage or external communication of this data beyond serving this single request. The specific design of mechanisms for controlled applications is left for deliverable D10.

5.3.5 Public Communication

As mentioned before, sometimes data needs to be communicated publicly. For example, applications relevant for vehicle and road safety require to send data in an unencrypted

form to other vehicles or backend services. In such a case, orthogonal mechanisms are required to conform with privacy policies. Techniques like data obfuscation or data fusion might need to be applied to fulfill the limited disclosure principle. This means that only the information that an application or user actually needs to perform a task should be sent. The needed information might actually vary throughout the system. For example, an exact position information sent in beacon messages might be needed in the vehicle domain for routing purposes or eSafety applications, while the same data when later being transmitted to a traffic operation center for traffic monitoring could also be used with much lower spatial resolution and thus a higher level of privacy.

We therefore assume that data obfuscation components are integrated in a system architecture wherever possible and useful. These data obfuscation components will perform one of the following operations on the data:

ID Removal: Removal of identifying data from the overall dataset, e.g., license plate information or cryptographic keys.

Spatial Cloaking: Reduction of spatial resolution of transmitted data, e.g., by adding random noise to the data, matching it to coarse-grained grid positions, and so forth.

Temporal Cloaking: Reduction of temporal resolution of transmitted data, e.g., by removing samples or by delaying transmissions.

Aggregation: Combination of different data samples into one data set e.g., by means of averaging, maximization, minimization, and so forth.

Note that there seems to be an obvious conflict between the requirements of the MPC Integrity Protection, that is, secured and encrypted transmission, and the need to modify data by the data cloaking component. However, these concepts are orthogonal. It needs to be decided on a per-application basis whether one-to-one-style communication is needed or unencrypted, broadcast-style communication mandates the need for data obfuscation techniques. While the exporter component supports both ways of communication, access to public communication from inside the PEP should be limited to a small fraction of controlled applications, e.g., specific safety applications. Both, public and confidential communication should support and employ pseudonymization of communication identifiers.

6 cITS Runtime Architecture - Reference Architecture

6.1 Overview

Chapter 5 introduced the concept of a Policy Enforcement Perimeter and major architecture building blocks which are necessary to ensure compliance at runtime with the privacy policies that have been introduced in Sections 3.2 and 4.2. In this Chapter, we outline in detail the PRECIOSA reference architecture, which realizes a Privacy-enforcing Runtime Architecture (PeRA) as envisioned in Chapter 5. The goal is an architecture that realizes the hippocratic principles, guarantees and enforces the adherence to user-specified policies, and provides system privacy throughout the whole cITS.

As outlined in Section 5.2, the PeRA needs to implement two major concepts to achieve these goals: Mandatory Privacy Control (MPC) and MPC Integrity Protection (MIP). MPC enforces compliance with privacy policies. The trusted state of MPC components is ensured by MIP. Also, MIP is required to establish trust in the distributed system of HcITS nodes. MIP also protects personal information during transit and storage.

Inside the Policy Enforcement Perimeter of the PRECIOSA PeRA, we assume a confidential communication channel between HcITS nodes, e.g., vehicles and backend servers. This way, personal information can be effectively tunneled through nodes and components that do not required access to it, e.g., forwarding nodes in the access domain. Section 6.2 elaborates on different communication paradigms supported by the PRECIOSA PeRA.

Figure 6.1 shows the components of the proposed PeRA implementation, including required subcomponents. It should be noted that some of those components may be optional, depending on the intended level of assurance. As discussed earlier, PRECIOSA-based systems require certification by a trusted party to confirm that proper privacy protection mechanisms are in place. It could be reasonable to introduce different levels of certification, depending on the assurance requirements demanded by data subjects.

Further, different supported paradigms for communication between cITS nodes are shown (see Section 6.2). The Privacy Control Monitor (PCM) is the core component of the architecture and consists of several subcomponents (see Section 6.3). The Privacy Policy Manager aides the PCM by managing policy storage, policy generation, and policy reasoning (see Section 6.4). Data and metadata is locally stored in a Secure Repository for which the PCM acts an access control point (see Section 6.5). Certain subcomponents are responsible for privacy maintenance (see Section 6.6). The Trust Manager is the local

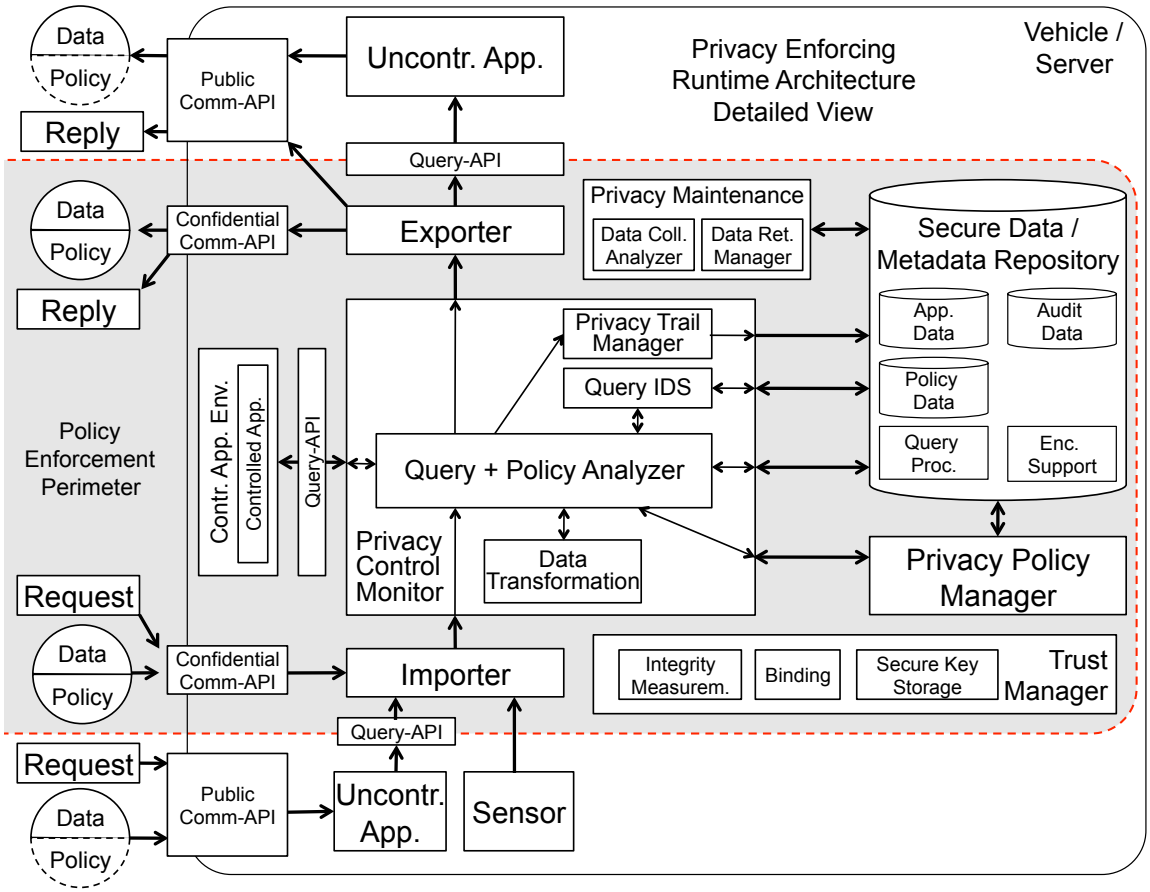

Figure 6.1: Privacy-enforcing runtime architecture – PRECIOSA reference architecture.

component responsible for controlling and ensuring that protected data is only accessible when the system is in a trusted state (see Section 6.7). Controlled applications can run inside the Policy Enforcement Perimeter in a controlled application environment (see Section 6.8). At the end of the Chapter, Section 6.9 discusses additional mechanisms to achieve communication privacy.

6.2 Communication Paradigms

The PeRA supports different communication paradigms inside its Policy Enforcement Perimeter. The *exporter* and *importer* components (see Figure 6.2) abstract from these paradigms and provide unified interfaces for incoming and outgoing communication to other PeRA components. Inside the Policy Enforcement Perimeter, confidential communication channels are used for data transfer between nodes. This is the assumed setting for the communication paradigms described below. Note, that some applications may not be able to utilize confidential communication, i.e., encrypt data in transit. For example, some

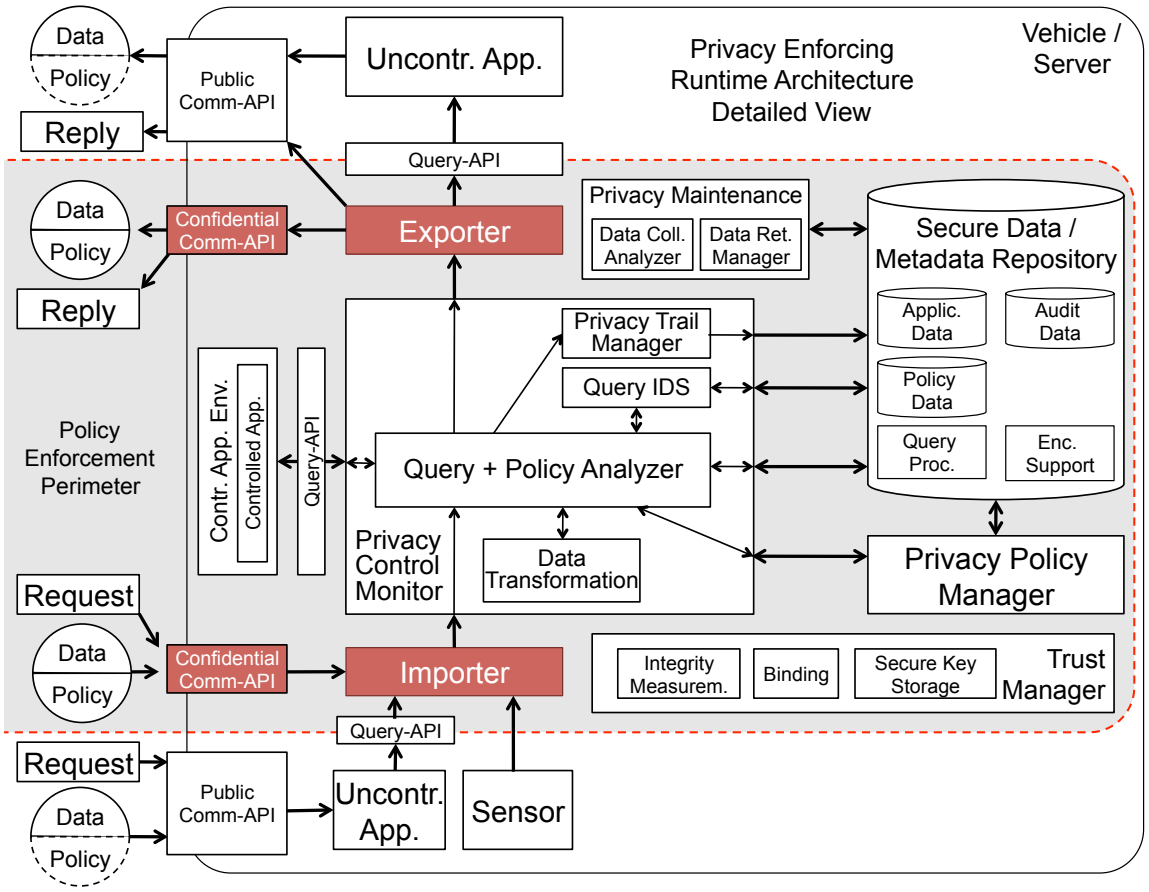

Figure 6.2: PeRA components: Components related to supported communication paradigms.

eSafety applications require dissemination of unencrypted data for direct consumption by other vehicles or for continuous re-broadcasting by RSUs. For these cases, additional privacy mechanisms are employed to ensure privacy in vehicle and access domain. Section 6.9 outlines dedicated privacy mechanisms for public communication.

6.2.1 Unidirectional Communication

In many ITS applications, vehicles submit data to backend servers in regular intervals. Examples are FCD or pay as you drive (PAYD) applications. These applications function only unidirectional and do not return any results to vehicles. When data is sent to the backend in this fashion, in our system, the importer component of the backend server is the communication endpoint inside the Policy Enforcement Perimeter. Messages are encrypted with the server's public key. The importer can decrypt the messages in cooperation with the Trust Manager and inserts data and attached metadata into the Secure Data/Metadata Repository.

Uncontrolled applications can then access the data through the query API. They pose a query request and specify a purpose for the request. The PCM ensures that only policy compliant data leaves the Policy Enforcement Perimeter. The PCM also attaches a merged privacy policy to the result set, which can, however, not be enforced by the PeRA anymore.

The importer also handles data that reaches a node from outside the Policy Enforcement Perimeter, e.g. FCD data form vehicles not equipped with the PRECIOSA architecture. If such data has privacy policies or meaningful metadata attached, they can be translated to PRECIOSA policies and enforced when processing the data inside the Policy Enforcement Perimeter. This way, the PeRA also supports the enforcement of privacy policies that have only been specified on a best effort level.

In a vehicle, to support FCD-style applications, local sensors are also exposed to the system through the importer. The application that creates the periodic reports in the FCD example is implemented as a restricted application in the vehicle. When sensor values are requested by the applications, the importer assigns default privacy policies to the sensor output according to user-defined privacy preferences. The controlled applications can bundle data in a packet and the PCM in cooperation with the Privacy Policy Manager assigns a merged privacy policy before it is send out via the exporter. How controlled applications function will be further discussed Section 6.8 and deliverable D10.

6.2.2 Bidirectional Communication

Some ITS applications require bidirectional transactions. For example, a vehicle requests up-to-date routing information for a certain road section. In such a case, the request as well as the server response may be privacy-sensitive. A privacy-sensitive request would be posed inside the Policy Enforcement Perimeter. Thus, the vehicle would attach a privacy policy to the request, the importer of the server recognizes the message as a request and passes it on to the appropriate restricted application. The application performs certain operations and creates a reply message. A merged policy, which takes into account the request policy and the polices of all other affected data items, is generated by the Privacy Policy Manager and attached to the reply by the PCM. A typical privacy policy could specify that a reply can only be sent to the original requester. Requests that are not privacy sensitive do not require their own policies and can be handled by uncontrolled applications on top of the PeRA.

6.3 Privacy Control Monitor

The concept of Mandatory Privacy Control as outlined in Section 5.2.1 is implemented in the reference architecture by the Privacy Control Monitor (PCM). It will ensure that privacy policies are respected on a mandatory level. To do this, all queries by applications have to be checked by the PCM. Results will only be returned if those are policy-compliant. Figure 6.3 shows the major components of the PCM.

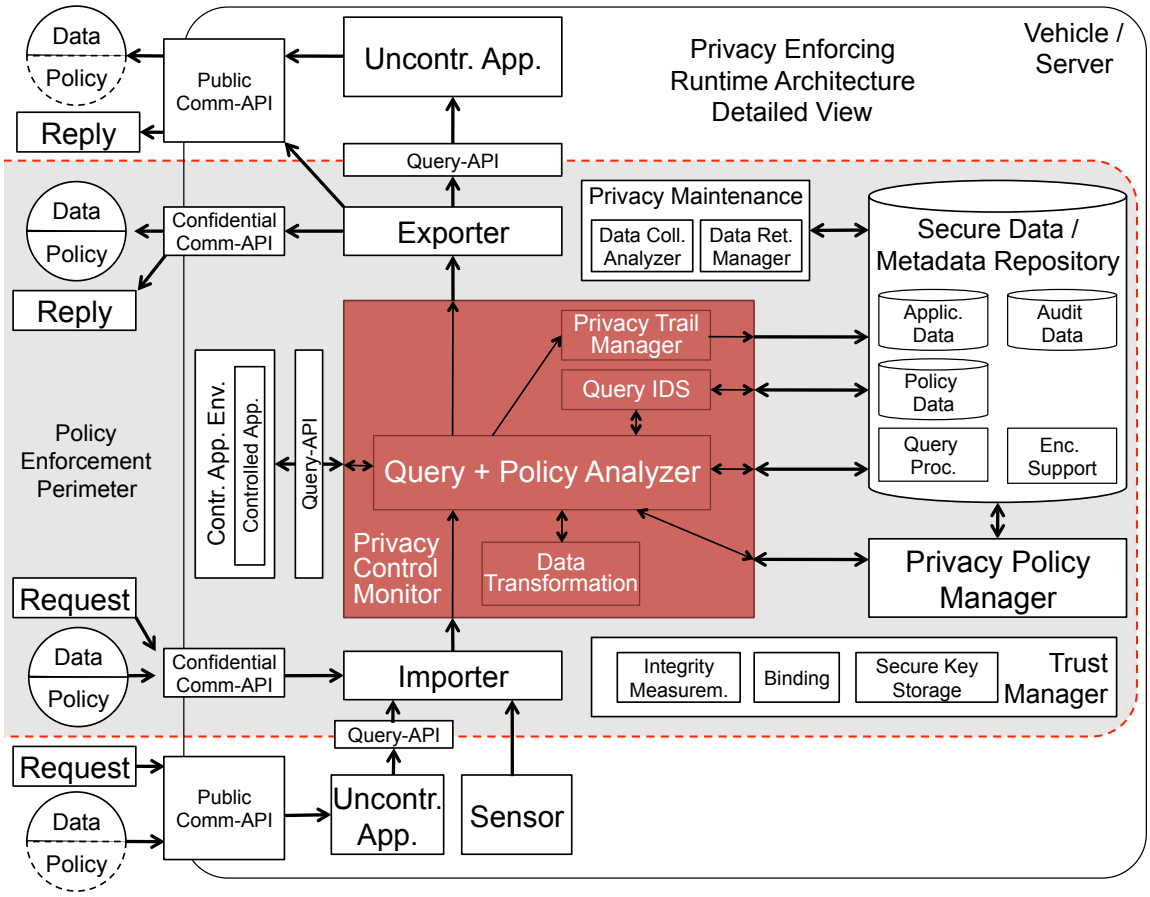

Figure 6.3: PeRA components: Privacy Control Monitor and its subcomponents.

Applications access data through a *query-API*, which is the interface of the PCM. A *Query and Policy Analyzer* inspects queries as well as results and ensures that they comply to the privacy policies of the requested data. An intrusion detection system monitors incoming queries (*Query IDS*) to detect potentially privacy infringing request patterns. All requests and operations are further logged by a *Privacy Trail Manager* to enable verifiability of policy compliance and, thereby, implement the principle of compliance. Specific *data transformations* can be performed by the PCM to comply to certain anonymization requirements and other restrictions specified in policies. For example, a privacy-aware average transformation could ensure the adherence to k-anonymity settings (cf. deliverable D2).

In the remainder of this section, we will describe the subcomponents of the PCM in detail. Note however, that we will stay on a conceptual level. Detailed specifications will be given in D10.

6.3.1 Query and Policy Analyzer

The purpose of the *Query and Policy Analyzer* is to analyze queries and results as well as their policy metadata provided by the Privacy Policy Manager. The aim of the analysis is to ensure that queries and results comply to the privacy policies defined for individual data items that contribute to a result or are touched by a query.

The Query and Policy Analyzer receives queries from applications via a query API. A generic query language could be SQL. Alternatively, a more application-specific query language can be used that allows higher level queries, e.g., a method to retrieve the traffic situation on certain road segments.

The following example highlights the functionality of the Query and Policy Analyzer. Assume that the data repository contains a table `traffic_reports` with columns `vehicle_id`, `position`, `speed`, and `road_segment`. Consider the following two SQL queries:

```
SELECT AVG(speed) FROM traffic_reports WHERE road_segment = 1345936
SELECT VID, position FROM traffic_reports WHERE road_segment = 1345936
```

In the first query, the application wants to know the average speed driven on a certain road segment identified by id `1345936`. The second query wants to retrieve individual positions of vehicles on that road segment.

Assume that the `traffic_reports` table contains the following data for road segment 1345936:

VID	Position	Speed	Road-Seg.
9567	9.546 / 49.234	25	1345936
4238	9.544 / 49.232	43	1345936
1932	9.547 / 49.235	53	1345936
3451	9.543 / 49.233	37	1345936
7823	9.545 / 49.234	49	1345936

Now, assume that Vehicle 3451 has set a policy that dictates that its data is only to be used in an aggregated way together with at least 4 other data values. The other vehicles have not set any policies that would restrict the usage of their data in this example.

In this case, the privacy policies of the data relevant for the first query can be satisfied and the PCM will allow the query. The result will be an average speed of 41.4 km/h. However, if vehicle 3451 would have set the minimum aggregation to 10, the anonymity set would not be large enough and the PCM would have to exclude this data item from the query, yielding an average speed of 42.5 km/h.

Note how data has been filtered from the returned result in this case. Application developers need to be aware of this possibility and keep it in mind when developing algorithms based on the returned data. Similarly, in case of the second query, the policy will prevent data from vehicle 3451 from being returned. Instead, only the VIDs and positions for the other four vehicles will be returned.

We assume that all query results that are policy-compliant can be made public and thus handed to uncontrolled applications. In the example, all vehicles besides 3451 that have not set an according privacy policy must be aware that the ITS system might make their data public. Of course, privacy laws still apply and could pose stronger restrictions on the data usage, even if a user has not set custom preferences. Such policies defined by the legislative can then be merged with user-defined policies by the Privacy Policy Manager to reflect the privacy preferences of a user and also comply with privacy and data retention laws. If an application needs to work on the individual position samples but the data should still be protected by PeRA, controlled applications, as presented in Section 6.8, should be used.

One should also note that sometimes individual data returned by a query will be policy-compliant, but could still mean a policy violation when being combined with results from several earlier or later queries. To keep the query analyzer reasonably simple, it will be designed stateless and not take this into consideration. Instead, the Intrusion Detection System (IDS) described in Section 6.3.3 is intended to discover and potentially prevent such sets of queries.

6.3.2 Privacy Trail Manager

The Privacy Trail Manager provides a central place for logging queries, requests, and performed operations, e.g., insertion or deletion of data items. This trail of all privacy-relevant operations is necessary for later audit operations, compliance checking, and the investigation of privacy incidents.

It is a requirement to protect the integrity of the privacy trail, i.e., a later modification of audit data should be prevented or at least detected. For example, deletion or modifications of privacy trail records can be detected if the trustworthy Privacy Trail Manager attaches a serial number or time stamp and digital signature to each record. Alternatively, one could write these records to tamper-resistant write-once (read-many) memory. As a third approach, it could – depending on the specific application – be possible to keep a trail of all accesses to a users' data locally with the users. Thereby, users are enabled to proof non-repudiation of their data when necessary.

6.3.3 Query IDS

The *Query Intrusion Detection System (Query IDS)* is responsible for monitoring incoming queries to identify those (possibly authorized) users that misuse the system or perform attacks to undermine the privacy protection provided by the PCM and the PeRA system as a whole.

For this purpose, the Query IDS passively monitors all queries passing through the PCM. Based on a history of queries and results, the IDS will take an anomaly-detection-based approach to find *unusual* query patterns. The Query IDS rates queries in terms of trustworthiness and triggers an alarm for queries below a certain rating threshold. The Query

and Policy Analyzer can react accordingly by blocking further queries belonging to the same set of queries, for example, all further queries originating from the same user or application. Anomalous queries might be defined by excessive rates of queries, queries that systematically probe through the database, or a high number of queries violating certain policies.

To achieve this goal, the Query IDS cooperates with the Privacy Trail Manager (see Section 6.3.2) and the Data Collection Analyzer (see Section 6.6.1). These components provide the information that the Query IDS requires to generate intrusion models, which are used to rate queries. What information needs to be collected for this purpose is data and application specific.

While the Query IDS is mainly concerned with on-line detection of privacy infringement and security breaches, the Data Collection Analyzer also monitors off-line operations on the collected data, i.e., maintenance operations and the enforcement of limited retention policies.

6.3.4 Data Transformations

The *data transformation* component provides an extensible set of transforming operations that can be applied to data by the PCM. Those transformations serve the purpose of fulfilling certain anonymization requirements posed by privacy policies of data items. Possible transformations are aggregation (summing up or averaging data items), k-anonymity, data cloaking mechanisms, or transformations related to other specific privacy mechanisms. Privacy policies have to be adapted accordingly before the result set is returned.

6.4 Privacy Policy Manager

As its name implies, the *Privacy Policy Manager* (see highlighted part in Figure 6.4) is responsible for handling privacy policies and performing policy related tasks. The general design of privacy policies has been discussed in Section 4.2 and will be further detailed in following deliverables.

At an information source, before any application can be used, the data subject must express its privacy preferences. The Privacy Policy Manager first translates these user-specified privacy preferences into privacy policies understood by the PeRA system. Then, it acts as a privacy constraint validator [4] and verifies that user privacy policies, service provider policies, and governmental privacy and data retention policies are all in accordance. For example, the service provider might express a policy to store data for audit reasons longer that the duration preferred by the data subject. Those conflicts must be detected, resolved if possible, and also be reported back to the data subject.

After validation, the Privacy Policy Manager generates corresponding metadata with its necessary structures and content that represents policies in a machine accessible form. Policies are assigned unique IDs to facilitate coupling of data and metadata, as well as

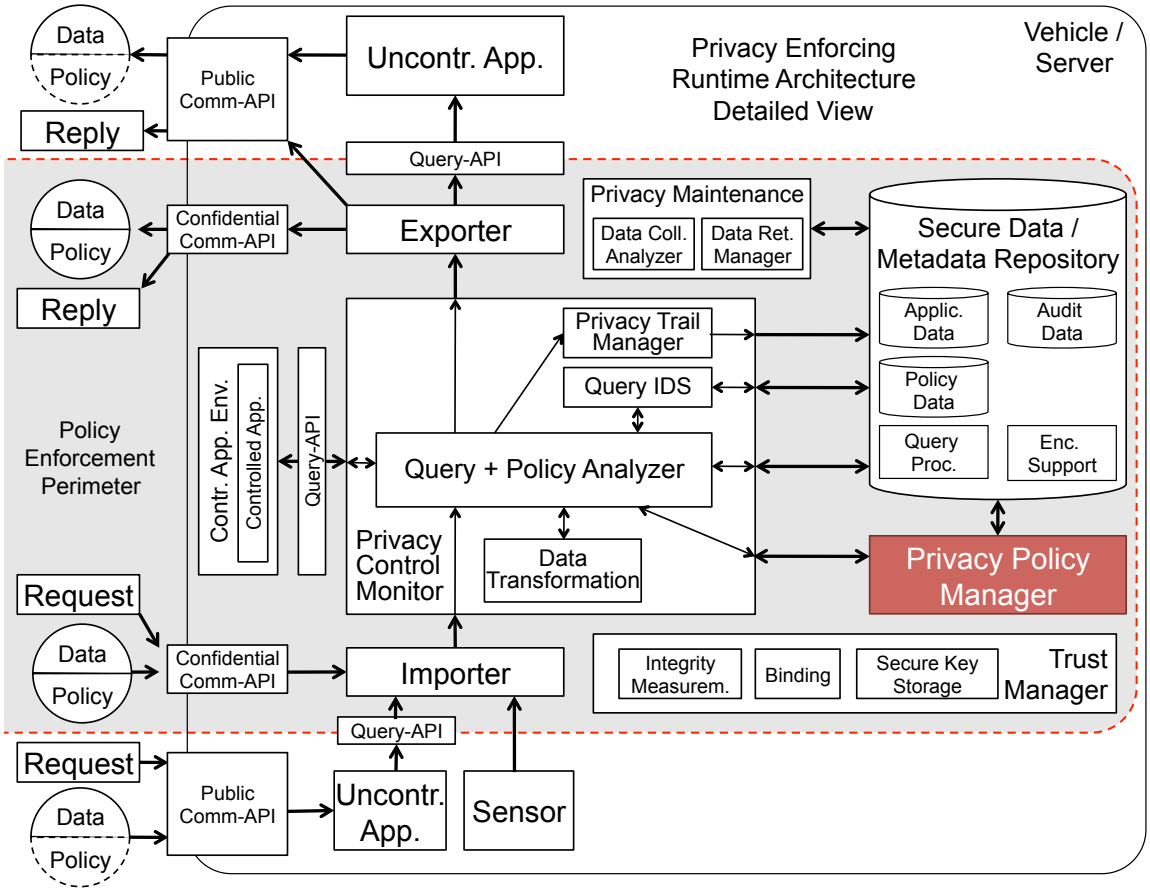

Figure 6.4: PeRA Components: Privacy Policy Manager.

efficient transmission and storage of policies. All polices data is stored in the Secure Data/Metadata Repository, but only the Privacy Policy Manager acts upon metadata.

When a query is being processed by the PCM, it is the Privacy Policy Manager's task to retrieve the policies of the affected data items and provide them to the PCM. This metadata can then be used by the policy and query analyzer to accept or reject queries. The Privacy Policy Manager also performs unification of multiple privacy policies resulting in a merged policy that corresponds to a result set. How policies are merged is policy and application specific.

6.5 Secure Data/Metadata Repository

At each HcITS node, personal information is stored in a *Secure Data/Metadata Repository* as shown in Figure 6.5. This repository is most likely an encrypted database which is only accessible through the PCM. Due to the MPC Integrity Protection and the Trust Manager, the database can only be decrypted if the relevant components, i.e., DBMS and PCM,

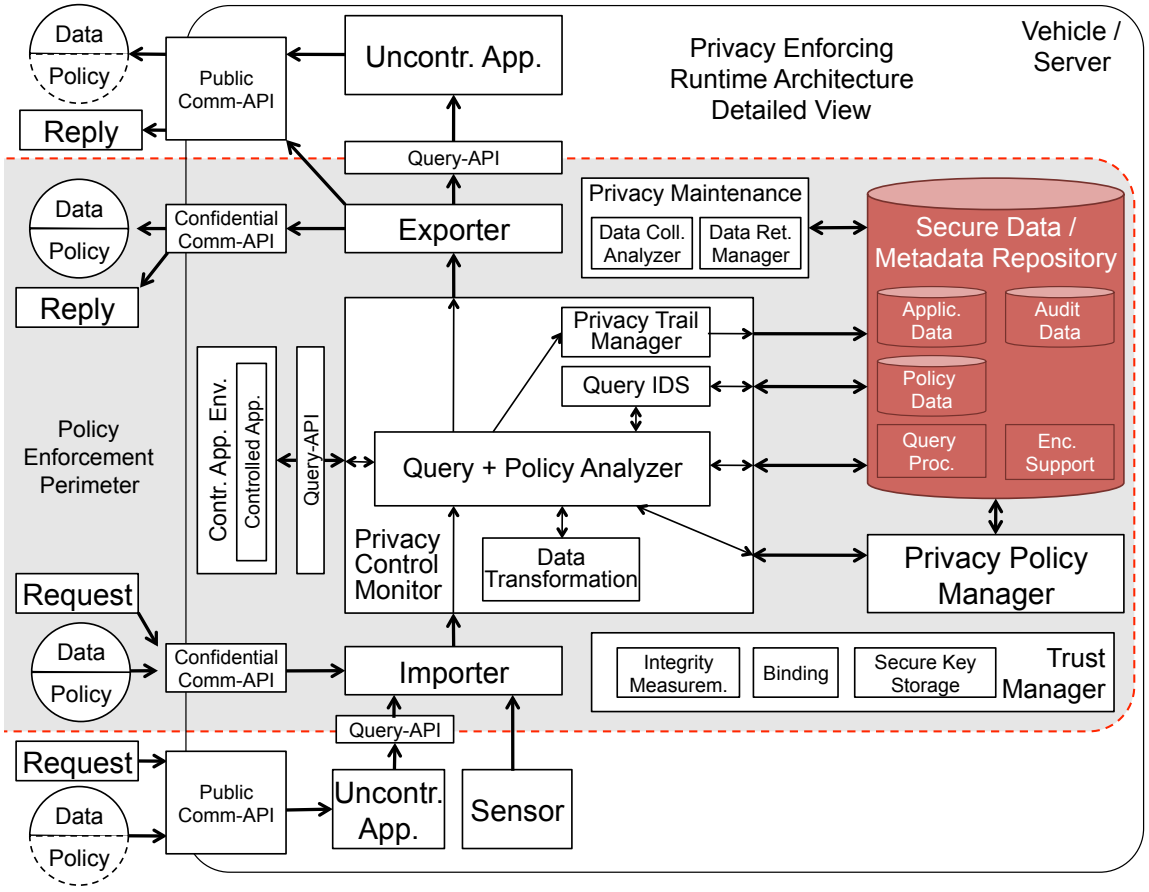

Figure 6.5: PeRA Components: Secure Data/Metdata Repository.

are in a trusted state. The query processor can either be seen as a part of the secure repository or as a subcomponent of the PCM. Which approach is chosen is implementation specific, and may be governed by efficiency considerations. Deliverable D10 will elaborate on this in more detail.

How database encryption is realized depends on the employed database management system and the underlying database engine, e.g., Berkeley DB [26]. Nevertheless, it can be assumed that symmetric encryption will be utilized in some form due to performance reasons. The required symmetric keys would be stored and managed by the Trust Manager, as described in 6.7. An issue that needs to be addressed in future deliverables in this context, is the replication of databases for backup or load balancing purposes.

The Secure Data/Metadata Repository holds three kinds of data: application data, policy data, and audit data.

6.5.1 Application data

Application data is all data that is not policy data or audit data, that is, application data is the *real* data, while policy and audit data are *metadata*. Application data can be data generated by the local node, e.g., sensor values that are stored or cached; data received from other HcITS nodes, e.g., floating car data donated by vehicles; or other data imported into the data repository.

Application data is the information that the PeRA primarily protects. Only the PCM can access and operate on application data.

6.5.2 Policy data

All policies and policy-related metadata are also stored in the secure repository to prevent tampering and modification of policies coupled to data. Policy data is only accessed and operated on by the Privacy Policy Manager.

6.5.3 Audit data

Audit logs generated by the Privacy Trail Manager also need to be stored in a secure manner. Preferably, this data should be stored in write-once, ready-many (WORM) memory to prevent tampering. The data collected by the Privacy Trail Manager is only of value as long as its integrity is guaranteed.

6.6 Privacy Maintenance

Privacy Maintenance (highlighted in Figure 6.6) encompasses all components that are not involved in on-line processing of queries. Components in this category ensure that the principles of limited retention and limited collection are followed. They perform certain off-line tasks for this purpose.

6.6.1 Data Collection Analyzer

In order to detect privacy or security breaches on-line or off-line, it is the *Data Collection Analyzer*'s responsibility to collect and analyze the sequence of events such as applications submitting a query or sending a message to another HcITS component. Which information needs to be collected is up for a specific design. While the Query IDS performs on-line query monitoring, the Data Collection Analyzer performs analysis over spatiotemporal data to provide input for the Query IDS.

Therefore, the Data Collection Analyzer examines operations either performed on stored data or on the communication link. Its goal is to

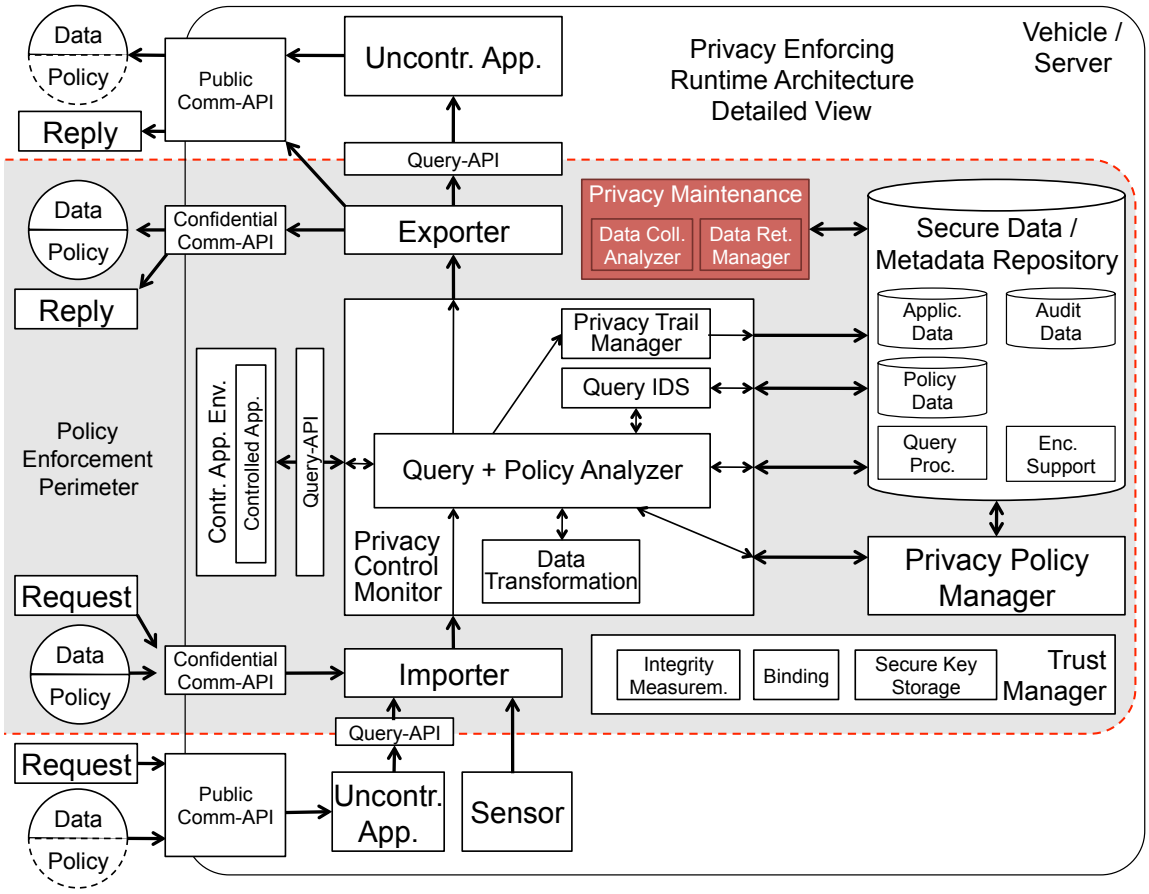

Figure 6.6: PeRA Components: Privacy maintenance subcomponents.

- control whether any information is being collected (i.e., stored or transmitted), but not used in order to support the principle of limited collection;
- analyze whether data is stored longer than necessary or specified by the data subject, thus supporting the principles of limited retention and limited use;
- collect data on applications that query information for detecting patterns that might suggest an intrusion of the HcITS component; and
- collect data for auditing purposes that are required by legal or governmental regulations; this can also mean to generate reports from data created by the Privacy Trail Manager.

6.6.2 Data Retention Manager

The *Data Retention Manager* ensures that data is removed from the data repository when it is not supposed to be used anymore according to the data subject's privacy policies. In

order to do this, it keeps a time-sorted queue of events that it will process to delete data at a certain time. Note that retention can not only be limited in time, but also to a geographic region. Furthermore, limited retention does not necessarily mean that data needs to be deleted completely. It could also mean that data may only be used as a single data item for a limited time period and afterwards only in aggregated or anonymized form.

We realize that it might be necessary to make data inaccessible (i.e., logically deleting it) rather than physically deleting it if governmental or legal requirements exist. for example, when the usage retention period of a data item expires, it could be *sealed*, i.e., encrypted, with the public key of some state or governmental authority and archived. How to manage this sealing of data such that it becomes inaccessible under regular terms will be elaborated on in detail in later deliverables.

Of course, data retention can only be guaranteed on cITS nodes that implement MPC and MIP. Therefore, data that leaves the Policy Enforcement Perimeter must be processed to be compliant with its policies and take this factor into account.

6.7 Trust Manager

MPC Integrity Protection, as introduced in Chapter 5, aims to protect data during transit and storage. This requires a trust relationship between vehicle and server, because the vehicle has to trust in the fact that privacy relevant information is only processed by trusted components of the PeRA and that these components are actually trustworthy and not controlled by an adversary. Thus, MIP also prevents lower layer attacks on MPC by ensuring the integrity of MPC components. It is the responsibility of the Trust Manager to realize MIP, see Figure 6.7. Trusted computing principles can be employed to achieve this.

The specific implementation of MIP and the Trust Manager strongly depends on the underlying platform, e.g., whether a Hardware Security Module (HSM) is available that can be used as an anchor of trust. For the rest of this text, we assume that such an HSM is available. However, implementations with weaker requirements (and likely weaker assurance) are possible.

Different requirements apply to such components when used in a server environment versus a vehicle: a server-side HSM needs to support a large number of parallel operation requests while it is much more important for a vehicle-based HSM that cryptographic operations are handled efficiently and that operations are energy efficient. Deliverable D10 will go into more detail in terms of MIP and Trust Manager implementation.

The Trust Manager enables remote nodes to establish trust in the local node. This complements the trust placed into the third party that certifies the PeRA running on a HcITS node as well as the certification process. The Trust Manager or underlying hardware components require integrity measurement capabilities to determine whether certain system components are in a trusted state. Data is only accessible when the system is in a trusted state. Key material also needs to be managed in a way that it cannot be compromised or leaked.

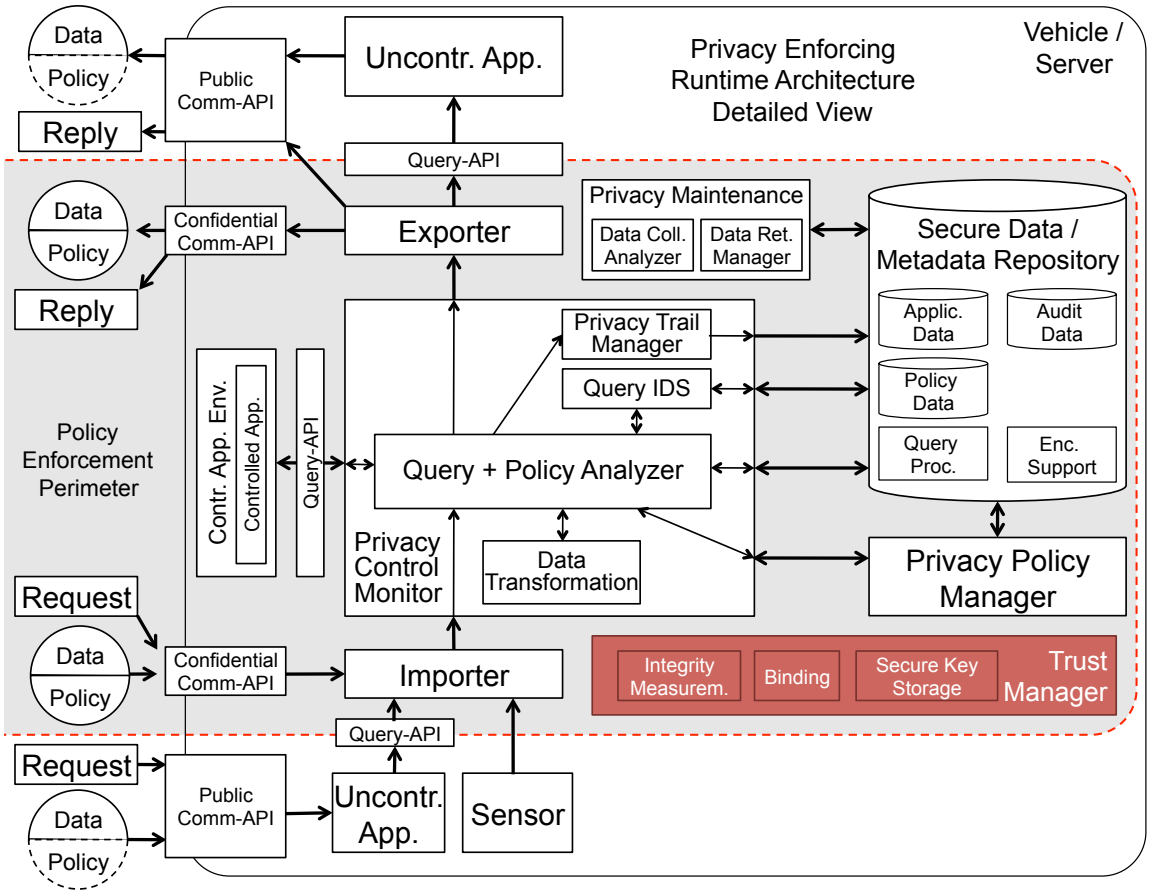

Figure 6.7: PeRA Components: Trust manager and its subcomponents.

6.7.1 Integrity Measurement

The Trust Manager acts as a trust anchor for remote platforms. This feature can be achieved in several ways, e.g., with a hardware module. The Trust Manager performs integrity measurement of other components and can assert their integrity. Integrity measurement hereby serves the purpose of determining whether a platform is in a trusted state. An integrity measurement component measures the integrity of other components before granting them access to certain functionality, e.g., access to the data repository. Dynamic integrity measurement allows to continuously monitor the integrity of certain components rather than only measuring their integrity on initial startup. Measurement results are stored as integrity digests, i.e., hash values, in platform configuration registers, which can only be extended but not overridden. Note that integrity measurement does neither prevent components nor the system from reaching an untrusted state, but it guarantees that the platform state is accurately measured and stored. Therefore, the Trust Manager has a sound basis for access decisions to the data repository and key material.

The integrity of several components of the PeRA needs to be measured and monitored. To be able to successfully enforce privacy, it is essential to guarantee that the PCM is in a trusted state. The same holds true for the state of the data/metadata repository, the Privacy Policy Manager, and the execution environment for controlled applications. In case one of these components is in an untrusted state, this component is denied access to any personal information or sensitive data. This could mean that incoming messages or the database cannot be decrypted, because the Trust Manager refuses cooperation. When the platform is restored to a trusted state, access would be granted again. Nevertheless, hardware and software updates are inevitable in some environments, especially in the case of backend servers. These issues and further details on how proper integrity measurement can be achieved in a system implementation will be discussed later in deliverables D10.

6.7.2 Binding and Sealing

The Trust Manager supports two asymmetric encryption variants: binding and sealing. This is similar to the Trusted Platform Module (TPM) Specification [27] published by the Trusted Computing Group.[1]

Binding refers to encryption in the traditional sense. Data is encrypted with a public key and can be decrypted with the corresponding private key without further restrictions. Sealing extends this concept by including information about the system state. Data is encrypted with a public key and against a set of platform configuration registers (PCR) and their values, so that decryption with the corresponding private key can only be performed if the specified PCRs contain the correct integrity digests. This is enforced by the Trust Manager or underlying hardware.

The use of sealing is straightforward. It ensures that data can only be decrypted (or unsealed) when the platform is in a trusted state, i.e., a state in which the integrity measurement of system components results in integrity digests that are known to be trustworthy. Sealing can be applied to arbitrary data and can, therefore, also be used to seal symmetric keys to a trusted state. Furthermore, the sealing concept can also be utilized to restrict the usage of asymmetric keys by specifying PCR indices and corresponding PCR values on key creation. Decryption operations with keys sealed this way can only be performed if the PCRs contain the correct state. Otherwise, key usage is denied by the Trust Manager.

For the PRECIOSA PeRA, we use sealing to enable confidential communication between HcITS nodes, e.g., vehicles and servers. An asymmetric key pair is initially created locally at an HcITS node by the Trust Manager and sealed to a set of PCRs comprising the desired trusted state of the platform. Subsequently, the public key of that key pair is certified by a trusted third party which asserts the privacy compliance of the HcITS node and verifies that the corresponding private key is only available in the trusted and certified

[1] The TCG is a not-for-profit organization formed by industry stakeholders to develop, define, and promote open standards for hardware-enabled trusted computing and security technologies with the self-proclaimed aim of helping users to protect their information assets (data, passwords, keys, etc.) from compromise due to attacks or physical theft. TCG website: www.trustedcomputinggroup.org

state. The resulting public key certificate can then be published and distributed to other HcITS nodes.

A remote HcITS node can encrypt messages with the public key contained in the certificate before sending them, most likely by encrypting the message with a random symmetric key which is then encrypted with the public key due to performance and message size considerations. This way, intermediate nodes are not able to decrypt the message. It is also possible to encrypt parts of the message for different recipients, e.g., a personal proxy may handle requests and replies for a vehicle, but would not be able to decrypt payment information destined for a billing service. At the receiving node, the Trust Manager decrypts the message if and only if the platform is in the state specified upon key creation and thus certified as privacy compliant by a third party.

Before data is prepared for sending, it first needs to be coupled with the metadata that describes corresponding privacy policies. This metadata has to be attached in such a way that it cannot be modified or decoupled from the data. For this, both data and metadata need to be signed by the originating vehicle.

6.7.3 Secure Key Storage

The Trust Manager is also responsible for managing cryptographic key material and its secure storage. This includes secret keys corresponding to certified public keys, as well as encryption keys for the data repository. The Trust Manager only allows use of decryption keys under its control when the system integrity and trusted state of the system have been verified. For example, if the data repository is in an untrusted state, the Trust Manager would not release the symmetric key the data repository is encrypted with, and it would remain sealed.

One way to securely store key material is placing it in a hardware security module. In that case, keys should also be flagged as non-migratable to ensure that they cannot be exported from the HSM that created them. However, if non-migratable keys are utilized, secure backup strategies are necessary to ensure system operation in case of failure of the HSM. These issues will be discussed in detail in later deliverables, namely D10 or D13.

Note that protection against techniques for key extraction from RAM [28] are currently out of scope. Several other research projects, like EMSCB[2] and openTC[3], already strive to enhance security of host platforms and to establish a fully trusted computing base on top of and complementary to the TCG specifications.

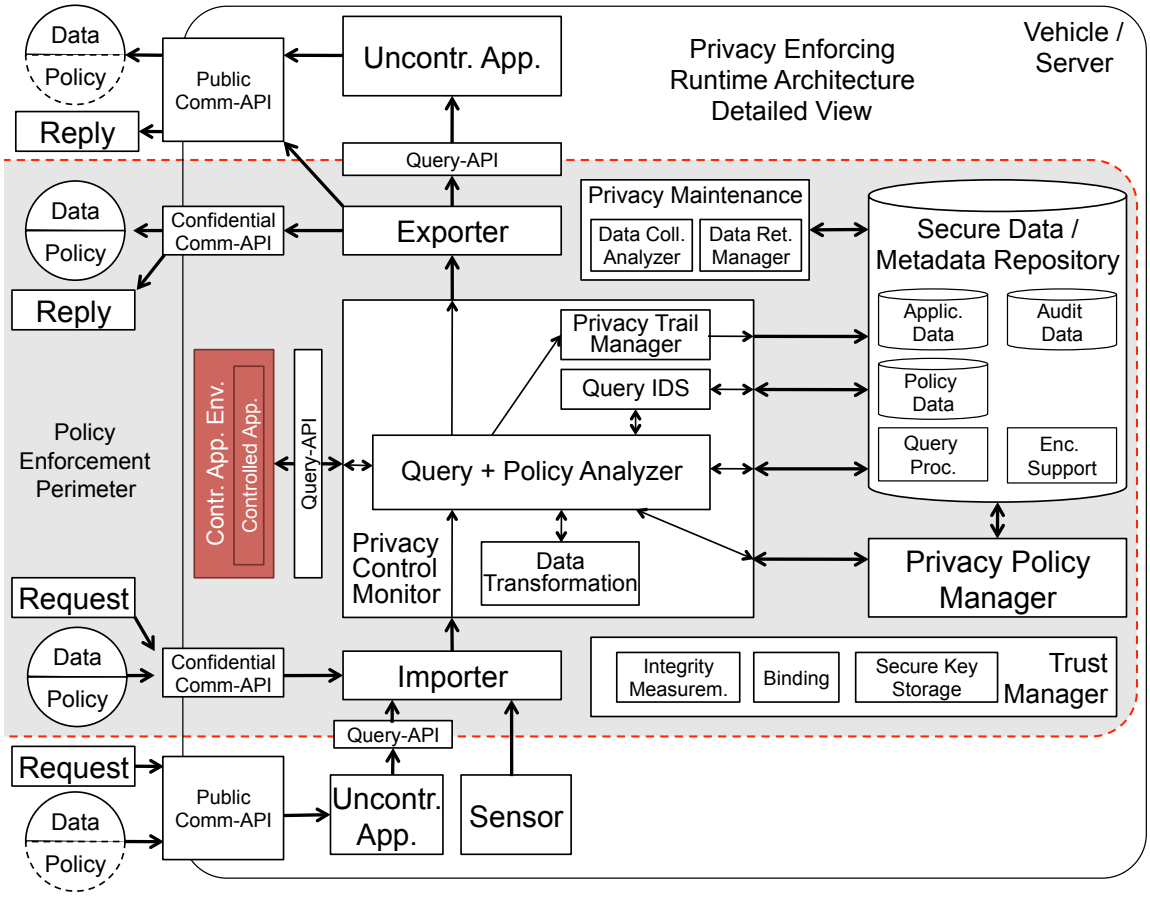

Figure 6.8: PeRA Components: Controlled applications and controlling components.

6.8 Controlled Applications

As mentioned before, some applications require direct interaction between HcITS nodes, e.g., a vehicle and a server. One scenario is that a server receives a request from a vehicle, performs some tasks, and returns a result to the vehicle. These request and result messages may be privacy sensitive. An example is a vehicle that wants to utilize up-to-date traffic information gathered from floating car data by a TOC server for navigation. The vehicle sends information about its current position and its destination to a TOC server requesting up-to-date traffic information for the roads on the way. It is obvious that the transferred positioning information is potentially privacy infringing. Similarly, when a server returns traffic information for a certain region, the destination of the vehicle could potentially be inferred from it.

[2]European Multilaterally Secure Computing Base (EMSCB) project website: http://www.emscb.com/
[3]Open Trusted Computing (openTC) porject website: http://www.opentc.net/

To allow this kind of interaction, an execution environment for controlled applications is provided inside the PeRA. This environment restricts the capabilities of an application and isolates it from other applications and the network. Importer and exporter act here as communication proxies between controlled applications and vehicles. This even allows to hide a vehicle's identity from the called restricted application.

6.8.1 Controlled Application Environment

The controlled application environment is an execution environment inside the PeRA for controlled applications. Capabilities of such applications are severely controlled to ensure that they cannot misuse privacy relevant information. That is, applications can only communicate with PeRA components, namely the query analyzer as an access point to the database and, tunneled through the PCM, the Importer and Exporter as communication proxies. Furthermore, controlled applications will not be able to use file system storage, they can only store information in the Secure Data/Metadata Repository of the PeRA.

Instances of controlled applications could not only be isolated from the outside world and from other applications but also from other instances of the same application. For each request, a new worker instance could be instantiated which performs its task and is terminated afterwards, thus making them stateless.

For some controlled applications, communication with outside services may be inevitable. The importer and exporter can be envisioned to act as privacy proxies that control the circumstances of such communication. In general, the capabilities of an application instance could be granted or restricted based on the privacy policy accompanying a given service request. This way, it is guaranteed that the execution environment is restricted to the degree demanded by the policy.

The introduction of controlled applications and the controlled application environment has the effect of logically shifting the application in question to the trust domain of the requesting node. For example, this way a vehicle can trust and use applications that should logically have been deployed in the vehicle, but require backend services (e.g., high-bandwidth access to the database) to function properly. Similar to the confidential channel which eliminates privacy threats from the access domain, the controlled application environment eliminates (some) privacy threats in the backend storage domain.

The controlled application environment could be enforced and implemented, for example, with the security framework of the Java language. The worker concept could be enhanced for performance by utilizing technologies like Enterprise Java Beans (EJB), OSGi, or Ice. Implementation details will be discussed in D10.

6.9 Mechanisms for Communication Privacy

Some applications require public communication to function properly, e.g., in broadcast scenarios. Therefore, orthogonal privacy mechanisms are required to ensure the user-

specified privacy level for such applications. Section 5.3.5 listed some general approaches in this direction which are extended in the following. We focus mainly on the vehicle domain, where such applications are envisioned for eSafety purposes.

As identified in D4, the characteristics of vehicular networks include high mobility of vehicles, restricted movements of vehicles, frequent broadcast of safety messages, no anonymous communicants, high demand of performance, high quality position information, and heterogeneous communication technologies. The wireless and open nature of vehicular networks exposes a set of points of attacks to both insider and outsider attackers (cf. Section 4.4.2). Put in a nutshell, the unique characteristics of vehicular communications mean that

- a substantial part of the communications, especially safety messages, cannot be encrypted and thus are sent in clear; and
- communications can be intercepted in the network, and personal information such as location information can be learned without reading the content of the messages.

Therefore, specific mechanisms for protecting communication privacy are needed to address such unique challenges in vehicle domain.

A multitude of mechanisms have been proposed in the past decades for preserving communication privacy. Most of the mechanisms are based on a few building blocks of communication privacy (e.g., pseudonyms and mix networks) and many of them are research proposals with limited practical value. D4 gives a comprehensive survey on the mechanisms for communication privacy and their interrelations.

The focus here is to choose mechanisms that can enforce privacy protection during communication in ITS. Besides, the mechanisms should meet the following two requirements:

1. The mechanisms should be applicable to a wide range of potential V2X applications.
2. The mechanisms should be practically implementable in ITS systems.

There will be a large amount of applications based on V2X communications. The applications encompass very different scenarios with diverse properties [29]. Consequently, the applications have very different communication patterns [30]. Thus, the mechanisms that we use in the architecture should be applicable for most, if not all, of the V2X applications. We also aim for a practical system in PRECIOSA, so the practicability of the mechanisms is also important.

The *pseudonym-based approach* developed by the SeVeCom project [31] meets both requirements and fits well into the architecture. Whenever identification is required in communicants, pseudonyms can be used instead of identifiers which can be linked to an individual. Pseudonymization can be used for public and confidential communication alike. The pseudonym-based approach is accepted by many projects and standardization organization in ITS, e.g., Network-on-wheels project [32], IEEE 1609.2 [33], and C2C-CC [34], to protect privacy in communications. Besides the design, SeVeCom has implemented a

prototype implementation of the mechanism on real V2X communication devices. A description of the design and implementation of the mechanism appears in [35] and [36], and we summarize the design of the mechanism in the following.

The pseudonym mechanism aims to provide privacy protection and to safeguard private information of the V2X system users. In the context of communication, the interest is in **anonymity** for the actions (messages and transactions) of the vehicles. The focus of the mechanism is on private vehicles (e.g., excluding emergency vehicles and buses), because the operation of all other V2X nodes, including RSUs, is not considered to raise any significant privacy concerns, and all those other nodes should be readily identifiable. A primary concern for cooperative ITS is to provide *location privacy*, that is, prevent others (any *observer*) from learning past or future locations of a VC system user (vehicle driver or passenger). To safeguard location privacy is to satisfy a more general requirement, i.e., anonymity for the vehicle message transmissions.

Ideally, it should be impossible for any observer to learn whether a specific vehicle transmitted a message or will transmit a message in the future (more generally, take an action). Further, it should be impossible to link any two or more messages (in general, actions) of the same vehicle. Even if an observer tried to guess, that should leave only a low probability of linking a vehicle's actions or identifying it among the set of all vehicles, the *anonymity set*.

Rather than aiming for this strong anonymity, the mechanism requires a relatively weaker level of protection: messages should not allow the identification of their sender, and two or more messages generated by the same vehicle should be difficult to link to each other. More precisely, messages produced by a vehicle over a protocol-selectable period of time τ can always be linked by an observer that received them. But messages m_1, m_2 generated at times t_1, t_2 such that $t_2 > t_1 + \tau$ cannot.

A fundamental support of the pseudonym mechanism is the Trusted Third Party (TTP) established in the backend system. The TTP is realized by instantiating *Certification Authorities* (CAs). Each CA is responsible for a *region* (e.g., national territory, district, or county) and manages identities and credentials of all nodes registered with it. To enable interactions between nodes from different regions, CAs provide certificates for other CAs (cross-certification) or provide *foreigner certificates* to vehicles that are register with another CA when they cross the geographical boundaries of their region

Each node is registered with only one CA, and has a unique *long-term* identity and a pair of *private* and *public* cryptographic keys, and it is equipped with a long-term *certificate*. A list of *node attributes* and a *lifetime* are included in the certificate, which the CA issues upon node registration. The CA is also responsible for the *eviction* of nodes and the *withdrawal* of compromised cryptographic keys via revocation of the corresponding certificates. In all cases, the interaction of nodes with the CA is infrequent and intermittent, with the roadside infrastructure acting as a gateway to and from the vehicular part of the network with the use of other infrastructure (e.g., cellular networks) also possible.

Each node X has a unique long-term identity ID_X, which will be the outcome of an agreement between car manufacturers and authorities, similar to the use of vehicle identification

numbers (VINs). Identifiers of the same format will be assigned both to vehicles and road-side units. Each identity is associated with a cryptographic key pair (SK_X, PK_X) and a set of attributes of node X. The attributes reflect technical characteristics of the node equipment (e.g., type, dimensions, sensors, and computing platform), as well as the role of the node in the system. Nodes can be, for example, private or public vehicles (buses), or vehicles with special characteristics (police patrol cars), or RSUs, with or without any special characteristics (offering connectivity to the Internet). The assignment of an identity, the selection of attributes appropriate for each node, and the generation of the certificate are performed "off-line," at the time the node is registered with the CA. The lifetime of the certificate is naturally long, following the node life-cycle (or a significant fraction of it).

To obtain pseudonyms, a vehicle V's Hardware Security Module (HSM) generates a set of key pairs $\{(SK_V^1, PK_V^1), \ldots, (SK_V^i, PK_V^i)\}$ and sends the public keys to a corresponding pseudonym provider (PP) via a secured communication channel. V utilizes its long-term identity ID_V to authenticate itself to the PP. The PP signs each of the public keys, PK_V^i, and generates a set of pseudonyms for V. Each pseudonym contains an identifier of the PP, the lifetime of the pseudonym, the public key, possibly an attribute set, and the signature of the PP; thus, no information about the identity of the vehicle is contained.

Pseudonyms are stored and managed in the on-board pseudonym pool, with their corresponding secret keys kept in the HSM. This ensures that each vehicle has exactly one key pair (own pseudonym and private key) that is active during each time period. Moreover, once the switch from the (SK_j, PK_j) to the $(j+1)$-st key pair (SK_{j+1}, PK_{j+1}) is done, no messages can be further signed with SK_j; even if the certificate for PK_j is not yet expired. In other words, pseudonymity cannot be abused: For example, a rogue vehicle cannot sign multiple beacons each with a different SK_j over a short period, and thus cannot appear as multiple vehicles.

A vehicle needs to contact the PP infrequently but regularly to obtain a new set of pseudonyms. For example, if a vehicle utilizes pseudonyms in set i, it obtains the $(i+1)$-st set of pseudonyms while it can still operate with the i-th set. It switches to the $(i+1)$-st set once no pseudonym in the i-th set can be used anymore. We term this process a *pseudonym refill*.

Due to the requirement for accountability, the PP archives the issued pseudonyms together with the vehicle's long-term identity. In case of an investigation, an authorized party can ask the PP to perform a *pseudonym resolution*: Reveal the link of a specific pseudonym to the long-term identity of the vehicle.

By using the same pseudonym only for a short period of time and switching to a new one, vehicle activities can only be linked over the period of using the same pseudonym. Changing pseudonyms makes it difficult for an adversary to link messages from the same vehicle and track its movements.

An adversary analyzing which certificates are attached to signed messages can track the location of vehicles over time. If pseudonyms are changed at appropriate time and location, messages signed under different pseudonyms are hard to be linked by an adversary.

As the adversary could use information from other layers of the communication stack to track vehicles (e.g., MAC- or IP-addresses), a change of pseudonyms should be accompanied by a change of the vehicle identifiers in underlying protocols as well. Still, using the location contained in messages to match pseudonyms, an adversary can indirectly identify vehicles by predicting the next position of a vehicle even if it has a new pseudonym. For some applications, cloaking of location information [37] is not a solution as it would e.g. jeopardize the use of safety applications. [38] proposes that vehicles change pseudonyms in regions not monitored by an adversary. These regions are called mix zones [39] as the vehicles by changing pseudonyms will mix with each other. Vehicles can also change their pseudonym at regular intervals maximizing the probability of changing a pseudonym in a mix zone. Another approach is proposed in [40] that creates unmonitored regions by encrypting communications (i.e., cryptographic mix zones) in small regions with the help of the road infrastructure.

When vehicles change pseudonyms in unmonitored regions of the network, mix zones are large and it is difficult for an adversary to obtain good estimations. However, when mix zones are created by the use of cryptography, they tend to be smaller and thus must be located appropriately to maximize their effectiveness (e.g., at traffic intersections). Hence, linking messages signed under different pseudonyms becomes increasingly hard over time and space for an adversary. As vehicles will change pseudonyms several times before reaching destination, the adversary will accumulate more uncertainty and like in mix networks [41], mobile nodes can achieve a high level of location privacy.

The SeVeCom project defines a baseline security architecture for VC systems [42, 35, 36]. Based on a set of design principles, SeVeCom defines an architecture that comprises different modules, each addressing certain security and privacy aspects. Modules contain components implementing one part of system functionality. The baseline specification provides one instantiation of the baseline architecture, building on well-established mechanisms and cryptographic primitives, thus being easy to implement and to deploy in upcoming VC systems.

The SeVeCom baseline architecture addresses different aspects, such as secure communication protocols, privacy protection, and in-vehicle security. As the design and development of VC protocols, system architectures, and security mechanisms is an ongoing process, only few parts of the overall system are yet finished or standardized. As a result, a VC security system cannot be based on a fixed platform but instead has to be flexible, with the possibility to adapt to future VC applications or new VC technologies.

To achieve the required flexibility, the SeVeCom baseline architecture consists of modules, which are responsible for a certain system aspect, such as identity management. The modules, in turn, are composed of multiple components each handling a specific task. For instance, the *secure communication module* is responsible for implementing protocols for secure communication and consists of several components, each of them implementing a single protocol. Components are instantiated only when their use is required by certain applications, and they use well-defined interfaces to communicate with other components. Thus, they can be exchanged by more recent versions, without other modules being affected.

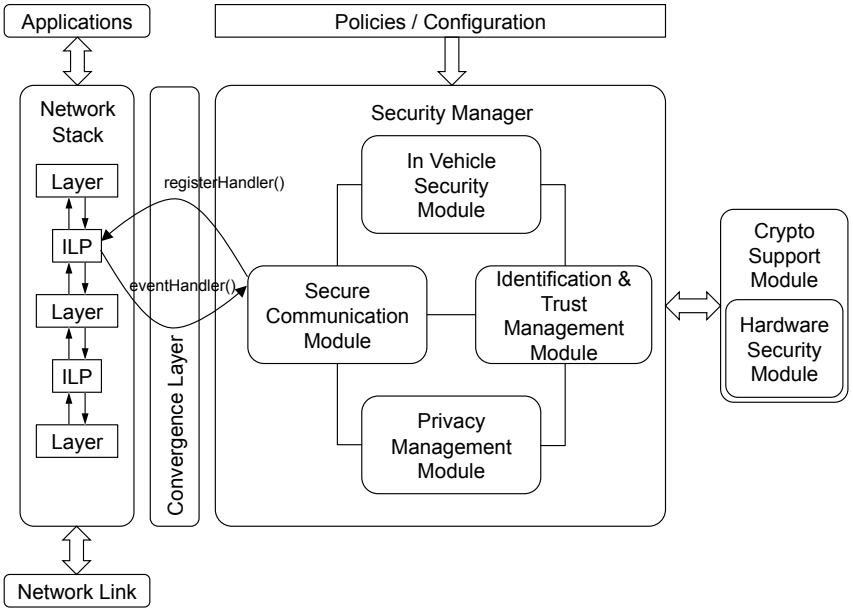

Figure 6.9: SeVeCom Baseline Architecture: Deployment View

As shown in Fig. 6.9, the *security manager* is the central part of the SeVeCom system architecture. It instantiates and configures the components of all other security modules and establishes the connection to the Cryptographic Support Module. To cope with different situations, the Security Manager maintains different policy sets. Policies can enable or disable some of the components or adjust their configuration, for example, to enhance or relax the parameters for a pseudonym change under certain circumstances.

To date, the SeVeCom baseline architecture can be considered as a relatively mature and practically deployable solution for security and privacy in vehicular communications. Nevertheless, ongoing research has mostly considered VC protocols relying on periodic beaconing, flooding, GeoCast, and position-based routing. Up to now, these mechanisms have received the attention of work on VC security and privacy. Nonetheless, recently, additional means of information dissemination have been considered in the context of VC. For example, the literature highlights the need for more efficient flooding and GeoCast strategies, and suggests the use of Gossiping or Context-adaptive Message Dissemination, as well as Data Aggregation in VC systems [30].

These new approaches will necessitate an adaptation of security and privacy strategies. Mechanisms such as context-adaptive message dissemination already provide an inherent degree of resistance against attacks [43]. In contrast to many routing protocols, where the protocol itself can become the target of an attacker, there is (nearly) no signaling between nodes that an attacker could exploit.

7 Summary, Conclusion, and Outlook

This document describes the idea and design of the so called “Privacy-verifiable Architecture”, i.e., an architecture that can guarantee certain privacy properties and these properties can be verified by some external partner, e.g., a user or trusted third party. It will guarantee that future ITS systems will be designed in a privacy friendly way and that users of such systems can rely on them to respect the privacy policies that the users set for their sensitive data.

The architecture is grounded on three building blocks:

1. The *Architecture Principles and Requirements* provide the basic ideas that the architecture framework is based upon.
2. The *Cooperative ITS Design Process* that provides guidelines and tools that cITS designers should use to follow a Privacy-by-Design strategy when building their cITS applications.
3. The *Cooperative ITS Runtime Architecture* that provides strong privacy guarantees through the Privacy-enforcing Runtime Architecture (PeRA). PeRA creates a policy enforcement perimeter where policies for personal information must be complied with in a mandatory way in order to access such information. PeRA implements two principle mechanisms: *Mandatory Privacy Control (MPC)* and *MPC Integrity Protection (MIP)* that will together ensure compliance with privacy policies.

Both the *Cooperative ITS Design Process* and the *Cooperative ITS Runtime Architecture* are each presented in two steps:

1. The *Cooperative ITS Architecture Framework* (AF) is a generic and reusable framework that substantiates the architecture principles.
2. The *Cooperative ITS Reference Architecture* provides a specific instantiation of the AF that will later be used to implement a prototype privacy-verifiable ITS based on the use cases from deliverable D1 [5].

In sum, this architecture will lift privacy in ITS (but also in generic IT systems) to a new level, as users of such systems remain in control of their private data and can rest assured that their policies will not be violated in the system.

While this document describes the overall architecture and outlines the major mechanisms, other deliverables give or will give a more detailed description of specific aspects. Namely, these are

D2 “V2X measurement approach”: describes measurement approaches that are used during the information flow analysis.

D6 “Models and privacy ontology for V2X”: provides the base for the policy language and guidelines.

D10 “Mechanisms for V2X Privacy”: is the direct continuation of this deliverable and will provide the details of various mechanisms that go beyond the scope of the architecture document.

D11 “Guidelines for privacy aware cooperative application”: Will provide the guidelines that are used during the design process.

D13 “V2X privacy mechanisms and verification support tools”: Will describe tool support during design- and run-time.

Finally, D14 “V2X privacy verifiable cooperative application” will provide a report on how a privacy-verifiable architecture has been designed and implemented based on the methodology and mechanisms described in this document.

8 Bibliography

[1] C2C Communication Consortium, “Car 2 car communication consortium manifesto v1.1,” online http://www.car-2-car.org/, August 2007.

[2] C. Project, “European its communication architecture – overall framework – proof of concept implementation,” online at http://http://www.comesafety.org/uploads/media/COMeSafety_DEL_D31_EuropeanITSCommunicationArchitecture_v2.0_01.pdf, October 2008.

[3] L. Cranor, “P3p: making privacy policies more useful,” *Security & Privacy, IEEE*, vol. 1, no. 6, pp. 50–55, Nov.-Dec. 2003.

[4] R. Agrawal, J. Kiernan, R. Srikant, and Y. Xu, “Hippocratic databases,” in *28th VLB Conference*, Hong Kong, China, 2002.

[5] PRECIOSA, “Deliverable 1: V2X privacy issue analysis.” [Online]. Available: http://www.preciosa-project.org/

[6] ISO TC 204/SC/WG 1, “Intelligent transport systems – system architecture – privacy aspects in its standards and systems,” ISO, Tech. Rep., 2008.

[7] APEC Secretariat, “Apec privacy framework,” Asian Pacific Economic Cooperation (APEC), Tech. Rep., 2005, ISBN: 981-05-4471-5.

[8] PRECIOSA, “Deliverable 4: Challenges for V2X privacy: issues.” [Online]. Available: http://www.preciosa-project.org/

[9] Wikipedia, “Data flow diagram — wikipedia, the free encyclopedia,” 2009, [Online; accessed 14-April-2009]. [Online]. Available: http://en.wikipedia.org/w/index.php?title=Data_flow_diagram&oldid=282974312

[10] G. Cutts, *Structured systems analysis and design methodology*. New York, NY, USA: Van Nostrand Reinhold Co., 1988.

[11] Wikipedia, “Activity diagram — wikipedia, the free encyclopedia,” 2009, [Online; accessed 14-April-2009]. [Online]. Available: http://en.wikipedia.org/w/index.php?title=Activity_diagram&oldid=277947665

[12] PRECIOSA, “Deliverable 2: V2X measurement approach.” [Online]. Available: http://www.preciosa-project.org/

[13] M. Backes, B. Köpf, and A. Rybalchenko, “Automatic Discovery and Quantification of Information Leaks,” in *Proc. 30th IEEE Symposium on Security and Privacy (S& P ’09), to appear*, 2009.

[14] I. Jacobson, G. Booch, and J. Rumbaugh, *The Unified Software Development Process*. Addison-Wesley, 1999.

[15] M. Raya and J.-P. Hubaux, "Securing vehicular ad hoc networks," *Journal of Computer Security (JCS), special issue on Security of Ad Hoc and Sensor Networks*, vol. volume 15, no. 1, pp. pages 39–68, January 2007.

[16] Wikipedia, "Man-in-the-middle attack — wikipedia, the free encyclopedia," 2009, [Online; accessed 2-February-2009]. [Online]. Available: http://en.wikipedia.org/w/index.php?title=Man-in-the-middle_attack&oldid=262943793

[17] M. Howard and D. LeBlanc, *Writing secure code*, 2nd ed. Redmond, WA, USA: Microsoft Press, 2003.

[18] B. Schneier, "Attack trees," *Dr. Dobb's Journal of Software Tools*, vol. 24, no. 12, pp. 21–22, 24, 26, 28–29, Dec. 1999.

[19] S. Mauw and M. Oostdijk, "Foundations of attack trees," in *ICISC*, ser. Lecture Notes in Computer Science, D. Won and S. Kim, Eds., vol. 3935. Springer, 2005, pp. 186–198. [Online]. Available: http://dx.doi.org/10.1007/11734727_17

[20] A. Aijaz, B. Bochow, F. Dötzer, A. Festag, M. Gerlach, R. Kroh, and T. Leinmüller, "Attacks on inter-vehicle communication systems - an analysis," in *3rd International Workshop on Intelligent Transportation (WIT 2006)*, March 2006.

[21] F. Swiderski and W. Snyder, *Threat modeling*. Microsoft Press, 2004.

[22] PRECIOSA, "Deliverable 6: Models and privacy ontology for V2X." [Online]. Available: http://www.preciosa-project.org/

[23] D. Bell and L. LaPadula, "Secure computer system unified exposition and multics interpretation," MITRE Corp., Bedford, MA, Tech. Rep. MTR-2997, July 1975.

[24] N. Ravi, M. Gruteser, and L. Iftode, "Non-inference: An information flow control model for location-based services," *Mobile and Ubiquitous Systems, Annual International Conference on*, vol. 0, pp. 1–10, 2006.

[25] S. Schlott, "Privacy- und Sicherheitsaspekte in ubiquitären Umgebungen," Dr. rer. nat., Faculty of Engineering and Computer Science, Ulm University, Albert Einstein Allee 11, 89081 Ulm, Germany, 2008, ISBN 978-3-89963-904-9.

[26] "Oracle Berkeley DB." [Online]. Available: http://www.oracle.com/technology/products/berkeley-db/index.html

[27] Trusted Computing Group, "TPM main specification," Trusted Computing Group, Main Specification Version 1.2 rev. 103, Jul. 2007. [Online]. Available: https://www.trustedcomputinggroup.org/specs/TPM/

[28] J. A. Halderman, S. D. Schoen, N. Heninger, W. Clarkson, W. Paul, J. A. Calandrino, A. J. Feldman, J. Appelbaum, and E. W. Felten, "Lest we remember: Cold boot attacks on encryption keys," in *USENIX Security Symposium*, P. C. van Oorschot, Ed. USENIX Association, 2008, pp. 45–60. [Online]. Available: http://www.usenix.org/events/sec08/tech/full_papers/halderman/halderman.pdf

[29] F. Kargl, Z. Ma, and E. Schoch, "Security engineering for vanets," in *4th Workshop on Embedded Security in Cars (escar 2006)*, November 2006.

[30] E. Schoch, F. Kargl, T. Leinmüller, and M. Weber, "Communication patterns in vanets," *IEEE Communications Magazine*, vol. 46, no. 11, p. 2–8, November 2008. [Online]. Available: http://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=4689254&isnumber=4689230

[31] "SEVECOM - Secure Vehicle Communications Project." [Online]. Available: http://www.sevecom.org/

[32] "NoW - Network on Wheels." [Online]. Available: http://www.network-on-wheels.de

[33] I. T. S. Committee, "Ieee 1609.2 trial-use standard for wireless access in vehicular environments-security services for applications and management messages."

[34] "Car2Car Communication Consortium." [Online]. Available: http://www.car-to-car.org/

[35] P. Papadimitratos, L. Buttyan, T. Holczer, E. Schoch, J. Freudiger, M. Raya, Z. Ma, F. Kargl, A. Kung, and J.-P. Hubaux, "Secure vehicular communications: Design and architecture," *IEEE Communications Magazine*, vol. 46, no. 11, p. 2–8, November 2008. [Online]. Available: http://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=4689252&isnumber=4689230

[36] F. Kargl, P. Papadimitratos, L. Buttyan, M. Müter, B. Wiedersheim, E. Schoch, T.-V. Tongh, G. Calandriello, A. Held, A. Kung, and J.-P. Hubaux, "Secure vehicular communications: Implementation, performance, and research challenges," *IEEE Communications Magazine*, vol. 46, no. 11, p. 2–8, November 2008. [Online]. Available: http://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=4689253&isnumber=4689230

[37] M. Gruteser and D. Grunwald, "Anonymous usage of location-based services through spatial and temporal cloaking," in *Proceedings of the ACM MobiSys*, New York, NY, USA, 2003, pp. 31 – 42.

[38] L. Buttyan, T. Holczer, and I. Vajda, "On the effectiveness of changing pseudonyms to provide location privacy in VANETs," in *ESAS 2007*, July 2007.

[39] A. R. Beresford and F. Stajano, "Location privacy in pervasive computing," *IEEE Pervasive Computing*, vol. 2, no. 1, pp. 46–55, 2003.

[40] J. Freudiger, M. Raya, M. Felegyhazi, P. Papadimitratos, and J.-P. Hubaux, "Mix-zones for location privacy in vehicular networks," in *WiN-ITS*, 2007.

[41] A. Serjantov and G. Danezis, "Towards an information theoretic metric for anonymity," in *Workshop on Privacy Enhancing Technologies*, 2002.

[42] SeVeCom, "Secure Vehicular Communications: Security Architecture and Mechanisms for V2V/V2I, Delivarable 2.1," http://www.sevecom.org, 2007-2008.

[43] E. Schoch, M. Keppler, F. Kargl, and M. Weber, "On the security of context-adaptive information dissemination," *Wiley Security and Communication Networks Journal*, vol. 1, no. 3, May/June 2008. [Online]. Available: http://www3.interscience.wiley.com/cgi-bin/fulltext/119139855/PDFSTART