# PREparing SEcuRe VEhicle-to-X Communication Systems

## Deliverable 1.3

## V2X Security Architecture v2

| | |
|---|---|
| Project: | PRESERVE |
| Project Number: | IST-269994 |
| Deliverable: | D1.3 |
| Title: | V2X Security Architecture v2 |
| Version: | 1.0 |
| Confidentiality: | Public |
| Editor: | Norbert Bißmeyer |
| Cont. Authors: | Norbert Bißmeyer, Sebastian Mauthofer, Jonathan Petit, Mirko Lange, Martin Moser, Daniel Estor, Michel Sall, Michael Feiri, Rim Moalla, Marcello Lagana, Frank Kargl |
| Date: | 2014-01-31 |

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

# Document History

| Version | Date | Main author | Summary of changes |
|---|---|---|---|
| v0.1 | 2013-09-09 | Norbert Bißmeyer (FhG SIT) | Initial structure of document created |
| v0.2 | 2013-11-01 | Sebastian Mauthofer (FhG SIT) | New structure of document created based on abstract on-board security architecture |
| v0.3 | 2013-11-15 | Norbert Bißmeyer (FhG SIT) | Description of abstract architecture added. Position of table headline changed. |
| v0.4 | 2013-11-27 | Norbert Bißmeyer (FhG SIT) | Section regarding meta data completed. Chapter 3 finalized. Figures in section 2.9.1 changed according to current ETSI GN and security standards |
| v0.5 | 2013-11-29 | Mirko Lange and Daniel Estor (Escrypt) | Information added to EVITA related sections. Review of infrastructure section. |
| v0.6 | 2013-12-05 | Jonathan Petit (UT) and Norbert Bißmeyer (FhG SIT) | Integration of OVERSEE into the VSS discussed in Section 2.4.2 (Secure SW) |
| v0.7 | 2013-12-23 | Michel Sall (Trialog) | Updates of sections on PMM, SCM, IDM and CRS |
| v0.8 | 2014-01-09 | Michael Feiri (UT) | Significantly reduced the level of detail of the CL API in the public architecture description |
| v0.9 | 2014-01-16 | Norbert Bißmeyer (FhG SIT) | Conclusion and summary added, Finalization of document. Preparation for internal review. |
| v0.10 | 2014-01-17 | Rim Moalla (Renault) | Contributions to Section 2.6 and 2.7. |
| v0.11 | 2014-01-27 | Frank Kargl (UT) | Complete revision. |
| v0.12 | 2014-01-28 | Sebastian Mauthofer (FhG SIT) | Complete revision. |
| v1.0 | 2014-01-30 | Norbert Bißmeyer (FhG SIT) | Finalization |

Table 1: Version history

| **Approval** | | |
|---|---|---|
| | Name | Date |
| Prepared | All Project Partners | 2014-01-17 |
| Reviewed | All Project Partners | 2014-01-27 |
| Authorized | All Project Partners | 2014-01-31 |

| **Circulation** | |
|---|---|
| Recipient | Date of submission |
| Project Partners | 2013-01-31 |
| European Commission | 2013-01-31 |

# Contents

# List of Figures

# List of Tables

# Glossary

| Abbrev | Synonyms | Description | Details |
|---|---|---|---|
| **API** | | Application Programming Interface | An API is a particular set of specifications that software programs can follow to communicate with each other. |
| **ASN.1** | | Abstract Syntax Notation One | ASN.1 is a standard and flexible notation that describes data structures for representing, encoding, transmitting, and decoding data. |
| **CA** | | Certificate Authority | A CA is an entity that issues digital certificates. |
| **CAM** | | Cooperative Awareness Message | CAMs are sent by vehicles multiple times a second (typically up to 10 Hz), they are broadcasted unencrypted over a single-hop and thus receivable by any receiver within range. They contain the vehicle's current position and speed, along with information such as steering wheel orientation, brake state, and vehicle length and width. |
| **CAN** | | Controller Area Network | A CAN is a vehicle bus standard designed to allow microcontrollers and on-board devices to communicate with each other. |
| **CCM** | | Communication Control Module | Module responsible for protecting on-board communication. Originates from the EVITA project. |
| **CCU** | | Communication & Control Unit | Hardware unit in an ITS station running the communication stack |
| **CL** | | Convergence Layer | Module that connects the external on-board entities (e.g. communication stack or applications) to the PRESERVE Vehicle Security Subsystem (VSS) |

| Abbrev | Synonyms | Description | Details |
|---|---|---|---|
| **CPU** | | Central Processing Unit | |
| **CRC** | | Cyclic Redundancy Code | Is used to produce a checksum in order to detect errors in data storage or transmission. |
| **CRS** | | Cryptographic Services | Module acting as proxy for accessing different cryptographic algorithm implementations. Originates from the EVITA project |
| **DoS** | | Denial of Service | A DoS is a form of attack on a computer system or networks. |
| **DENM** | DNM | Decentralized Environmental Notification Message | A DENM transmission is triggered by a cooperative road hazard warning application, providing information to other ITS stations about a specific driving environment event or traffic event. The ITS station that receives the DENM is able to provide appropriate HMI information to the end user, who makes use of these information or takes actions in its driving and traveling. Fehler: Referenz nicht gefunden |
| **EAM** | | Entity Authentication Module | Module responsible for ensuring entity authentication of on-board components. Originates from the EVITA project |
| **ECC** | | Elliptic Curve Cryptography | ECC is an approach to public-key cryptography based on the algebraic structure of elliptic curves over finite fields. |
| **ECU** | | Electronic Control Unit | |
| **FOT** | | Field Operational Test | |
| **G5A** | ITS-G5A | ITS road safety communication (802.11p) | Frequency band between 5.875 GHz and 5.905 GHz - reserved for ITS road safety communication |
| **G5B** | ITS-G5B | ITS non-safety communication (802.11p) | Frequency band between 5.855 GHz and 5.875 GHz - reserved for ITS road non-safety communication |
| **G5C** | ITS-G5C, C-WLAN | 5GHz wireless communication (802.11a) | |

| Abbrev | Synonyms | Description | Details |
|---|---|---|---|
| **GNSS** | GPS | Global Navigation Satellite System | Generic term for an Global navigation satellite system (GPS, GLONAS, Galileo) |
| **HMI** | | Human-Machine Interface | |
| **HSM** | | Hardware Security Module | |
| **I2V** | I2C | Infrastructure-to-Vehicle | Communication between infrastructure components like roadside units and vehicles |
| **I2I** | | Infrastructure-to-Infrastructure | Communication between multiple infrastructure components like roadside units |
| **ICS** | | ITS Central Station | ITS station in a central ITS subsystem |
| **IDK** | Module Authentication Key | Device Identity Key | The Device Identity Key is introduced by EVITA and is used for HSM identification. The IDK can also be certified by a manufacturer authentication key. |
| **IMT** | GSM, GPRS, UMTS | Public cellular services (2G, 3G, ...) | |
| **IPR** | | Intellectual Property Right | |
| **ITS** | | Intelligent Transportation Systems | Intelligent Transport Systems (ITS) are systems to support transportation of goods and humans with information and communication technologies in order to efficiently and safely use the transport infrastructure and transport means (cars, trains, planes, ships). |
| **ITS-S** | | ITS Station | Generic term for any ITS station like vehicle station, roadside unit, ... |
| **IDM** | | ID & Trust Management Module | Module responsible for ID management originating from SeVeCom project. |
| **IVS** | OBU | ITS Vehicle Station | The term "vehicle" can also be used within PRESERVE |

| Abbrev | Synonyms | Description | Details |
|---|---|---|---|
| **LDM** | Environment Table | Local Dynamic Map | Local geo-referenced database containing a V2X-relevant image of the real world |
| **LTC** | | Long-Term Certificate | PRESERVE realization of an ETSI Enrolment Credential. The long-term certificate authenticates a stations within the PKI, e.g., for PC refill and may contain identification data and properties. |
| **LTCA** | | Long-Term Certificate Authority | PRESERVE realization of an ETSI Enrollment Credential Authority that is part of the PKI and responsible for issuing long-term certificates. |
| **MAC** | | Media Access Control | The MAC data communication protocol sub-layer is a sublayer of the Data Link Layer specified in the seven-layer OSI model. |
| **OBU** | IVS | On-Board Unit | An OBU is part of the V2X communication system at an ITS station. In different implementations different devices are used (e.g. CCU and application unit) |
| **PAP** | | Policy Administration Point | Module related to the PDM originating from EVITA project |
| **PC** | Short Term Certificate | Pseudonym Certificate | A short term certificate authenticates stations in ITS-G5A communication and contains data reduced to a minimum. |
| **PCA** | | Pseudonym Certificate Authority | Certificate authority entity in the PKI that issues pseudonym certificates |
| **PCB** | | Printed Circuit Board | Board where circuits, e.g. micro controller, storage devices are placed and interconnected by conducting paths |
| **PDM** | | Policy Decision Module | Module responsible for enforcing the use of policies originating from EVITA project |
| **PDP** | | Policy Decision Point | Module related to the Policy Decision Module originating from EVITA project |

| Abbrev | Synonyms | Description | Details |
|---|---|---|---|
| **PeRA** | | Privacy-enforcing Runtime Architecture | Module responsible for enforcing privacy protection policies originating from PRECIOSA project |
| **PEP** | | Policy Enforcement Point | Module related to the Policy Decision Module originating from EVITA project |
| **PIM** | | Platform Integrity Module | Module responsible for ensuring on-board component integrity originating from EVITA project |
| **PKI** | | Public Key Infrastructure | A PKI is a set of hardware, software, policies, and procedures needed to create, manage, distribute, use, store, and revoke digital certificates. |
| **PMM** | | Pseudonym Management Module | Module responsible for management of the station's pseudonym certificates originating from SeVeCom project |
| **RSU** | IRS, ITS Roadside Station | Roadside Unit | A RSU is a stationary or mobile ITS station at the roadside acting as access point to the infrastructure. |
| **SAP** | | Service Access Point | Informative functional specification that enables the interconnection of different component implementations. |
| **SM** | | Security Manager | Module responsible for securing the V2X communication with external ITS stations originating from SeVeCom project |
| **SCM** | | Secure Communication Module | A generic name for the complete secure communication stack |
| **SEP** | | Security Event Processor | Module responsible for security event management (e.g. checking message plausibility, station reputation calculation) |
| **TPM** | | Trusted Platform Module | A TPM is both, the name of a published specification detailing a secure crypto-processor that can store cryptographic keys, as well as the general name of implementations of that specification, often called the "TPM chip" or "TPM Security Device". |

| Abbrev | Synonyms | Description | Details |
|---|---|---|---|
| **TRNG** | | True Random Number Generator | A hardware random number generator is an apparatus that generates random numbers from a physical process, rather than a computer program. |
| **V2I** | C2I | Vehicle-to-Infrastructure | Direct vehicle to roadside infrastructure communication using a wireless local area network |
| **V2V** | C2C | Vehicle-to-Vehicle | Direct vehicle(s) to vehicle(s) communication using a wireless local area network |
| **V2X** | C2X | Vehicle-to-Vehicle (V2V) and/or Vehicle-to-Infrastructure (V2I) | Direct vehicle(s) to vehicle(s) or vehicle(s) to infrastructure communication using a wireless local area network |
| **VSA** | | Vehicle Security Architecture | General outcome of PRESERVE work package 1 |
| **VSS** | | V2X Security Subsystem | Close-to-market implementation of the PRESERVE VSA that is the outcome of PRESERVE work package 2 |

# 1 Introduction

Protection of V2X communication in Intelligent Transportation Systems (ITS) against threats, as identified and addressed in the PRESERVE deliverable D1.1 [36], is very important for reliability and trustworthiness of such a communication. Most relevant threats described in the risk analysis of the PRESERVE deliverable D1.1 [36, Section 2.3] must be considered in order to define appropriate countermeasures regarding the information security features: *authenticity and authorization*, *availability*, *integrity*, *confidentiality*, *privacy* and *accountability*. An early integration of security and privacy protection mechanisms into the development process of V2X communication is necessary to consider early relevant threats and avoid possible vulnerabilities. In order to evaluate the security concepts in an early stage, it is reasonable to implement a close-to-market security solution in Field Operational Tests (FOTs).

This document describes the integrated Vehicle Security Architecture (VSA) of the PRESERVE project. It considers all relevant countermeasures from PRESERVE D1.1 in order to create a V2X security framework that can be used in different FOT environments. This architecture aims to be the basis for later implementation and operation within the project. It also builds upon PRESERVE Deliverable D1.2 and refines the discussions and results there by incorporating the intermediate project experiences and especially the results of the joint PRESERVE / C2C-CC Security Architecture Workshop held in June 2013 into a consistent architecture document. The VSA is designed to protect on the one hand the V2X communication between ITS stations and the on-board communication system by using results and solutions from the previous projects SeVeCom [25] and EVITA [40]. On the other hand, privacy protection is a very important aspect in V2X communications and therefore relevant mechanisms are considered in related processes. They are based primarily on results of the projects SeVeCom and PRECIOSA [24]. Furthermore, the VSA aims at being compatible with specifications defined by standardization bodies such as the European Telecommunications Standards Institute (ETSI), the Institute of Electrical and Electronics Engineers (IEEE) and the industrial driven consortium Car-to-Car Communication Consortium (C2C-CC). In order to be practically relevant for close-to-market FOTs, operational and evolutionary aspects such as re-usability, adaptability, scalability, and cost-effectiveness are considered by the VSA. Based on this VSA a V2X Security Subsystem (VSS) is created that combines different and partially enhanced security and privacy mechanisms from previous projects. The PRESERVE VSS aims to be usable in future V2X communication system implementations.

## 1.1 V2X Communication Network Architecture

Main participants of the V2X communication network are vehicles and the roadside facilities as depicted in Figure 1.1. The access points at the roadside act as gateways between the vehicles and backend services (e.g. central traffic management or fleet management) and additionally support multi-hop packet routing between distant vehicles. Access to cellular networks that may be used by vehicles to communicate with backend services are not assumed to be available in all vehicles. The VSA described in this document focuses on three participants in ITS communications: *vehicle station*, *roadside station*, and *central station* as depicted in Figure 1.1.

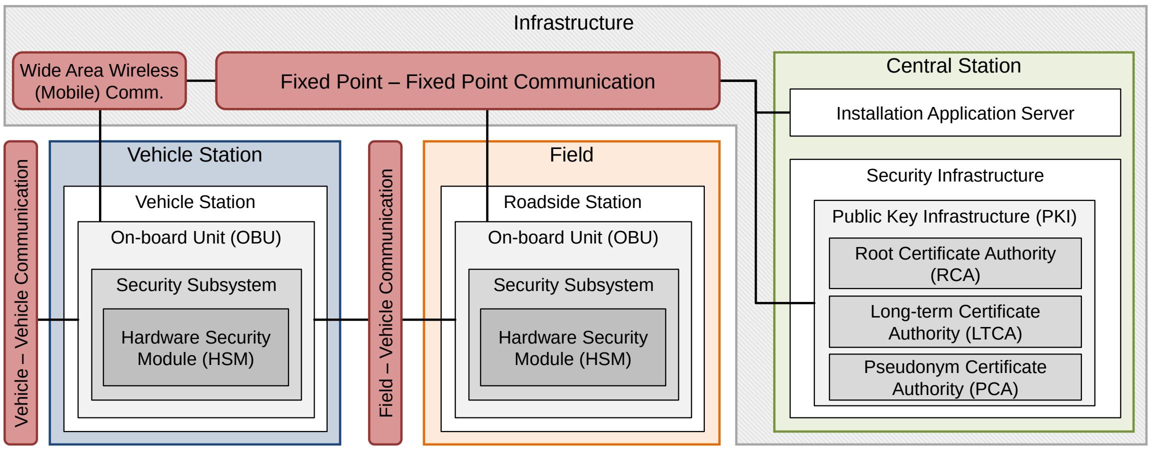

Infrastructure
Wide Area Wireless (Mobile) Comm.
Fixed Point – Fixed Point Communication
Central Station
Installation Application Server
Security Infrastructure
Public Key Infrastructure (PKI)
Root Certificate Authority (RCA)
Long-term Certificate Authority (LTCA)
Pseudonym Certificate Authority (PCA)
Vehicle – Vehicle Communication
Vehicle Station
Vehicle Station
On-board Unit (OBU)
Security Subsystem
Hardware Security Module (HSM)
Field – Vehicle Communication
Field
Roadside Station
On-board Unit (OBU)
Security Subsystem
Hardware Security Module (HSM)

Figure 1.1: Architecture of the Intelligent Transportation System based on [5, 30] illustrating relevant participants and communication channels

The representation of participants and communication channels in this figure is based on the description of the Intelligent Transportation System (ITS) architecture provided by the U.S. Department of Transportation [30] and ETSI [5]. Since these participants form a network with the depicted communication channels the participants are further named *station* of the ITS and are abbreviated in this document as ITS-S. In the following listing, the main participants of the network are discussed including their most relevant components.

- **Vehicle Stations** consist of an On-Board Unit (OBU) that is running the V2X applications, the communication facilities (i.e. radio, communication stack, etc.) and connects to the on-board network. The security subsystem of the station is connected to the OBU or comes as part of it. The subsystem provides security services to protect the on-board communication and the external V2X communication. A Hardware Security Module (HSM) is used in the security subsystem to store cryptographic credentials (i.e. private keys) and accelerate cryptographic operations. In parallel it acts as a trust anchor.
- In the **Field**, the most important participants are Roadside Stations:
  - The **Roadside Station**, also known as **Roadside Unit (RSU)**, consists of the same components as a vehicle station (i.e. OBU, security subsystem, HSM).

The roadside station is able to act as gateway between the vehicle communication and fixed point communication.

- The **Central Stations** provide the backend services. This deliverable focuses on the Installation Application Server and the Security Infrastructure:
  - The **Installation Application Server** provides software for vehicle stations and roadside stations (i.e. OBU and security subsystem). Possible operators of the server may be vehicle manufacturers or suppliers. The server is able to communicate with vehicles via wide area wireless communications (e.g. UMTS, LTE) or via fixed point entities such as RSUs.
  - The **Security Infrastructure** in the backend is running a security credential provider such as a **Public Key Infrastructure (PKI)** that is used to protect the V2X communication against external attackers. The PKI consist of different **Certificate Authorities (CAs)**. The purposes and tasks of the **Root CA (RCA)**, **Long-term CA (LTCA)**, and **Pseudonym CA (PCA)** are further detailed in Chapter 3. The security infrastructure is connected to the vehicles via fixed point communications or wide area wireless mobile communications.

According to Figure 1.1 different communication channels are used in ITS communications. However, this deliverable focuses on the wireless ad-hoc data transmission between vehicles (V2V) and between vehicles and the infrastructure (V2I). This kind of communication is further denoted as **V2X**. It is based on the IEEE standard 802.11p [23] and the European profile standard for ITS operating in the 5 GHz frequency band [6].

## 1.2 V2X Security Concepts

The ITS communication architecture described by ETSI [5] considers different communication systems dedicated to transportation scenarios. The general scenario described by ETSI consists of ITS domain specific elements (e.g. ITS-G5 or V2X message formats) and generic domain elements (e.g. GNSS or cellular networks). A general description of the assumed ITS environment can be found in the introduction of the PRESERVE deliverable D1.1 [36].

The PRESERVE security solution focuses on elements of the aforementioned ITS domain that are described in the following. The V2X security architecture aims at protecting primarily V2X ad-hoc communication based on ITS-G5A with their specific message types such as CAM [8] and DENM [9]. These V2X message types contain information required to enhance the road safety and efficiency. In order to ensure that vehicles and RSUs receive only consistent and unmanipulated data about their environment this road traffic related information needs to be secured. Furthermore, single-hop broadcast and unicast packet exchange between ITS stations via IEEE 802.11p and ITS-G5A will be in the main focus of this VSA. The protection of multi-hop packet forwarding is considered but not discussed in detail within this document. Likewise, session-based communication, i.e. via

Internet Protocol (IPv6), is not described in this document but the security architecture is extensible in order to protect this type of communication in a later stage.

The on-board V2X security solution of PRESERVE is shown in Figure 1.2. The VSS is placed inside the ITS station and aims to provide necessary security services as defined by ETSI in the ITS station reference architecture [5]. The VSS is connected to the on-board networks, the application unit that runs the V2X applications and the V2X communication entity that is connected with the outside world (i.e. ITS-G5A network).

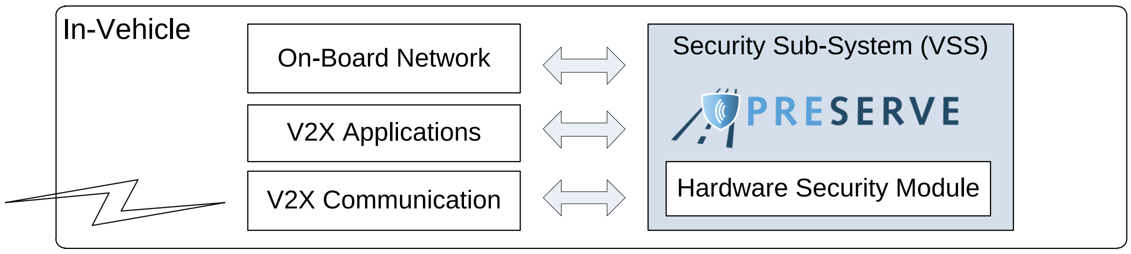


Figure 1.2: On-board V2X Security Solution

The VSS is therefore used to protect the V2X communication between ITS stations. In this concept digital certificates are used to sign outgoing packages that can afterwards be verified by the receiver's VSS. Main security operations are the signing and verification of broadcasted V2X messages. In addition the encryption and decryption of unicast messages is also considered by the VSA.

As stated in the PRESERVE deliverable D1.1 [36], the ad-hoc communication link based on ITS-G5A is assumed to be unstable due to given channel properties and frequently changing communication endpoints. Furthermore, in most cases new ITS stations that enter the communication range of another ITS station are unknown. In order to ensure trustworthy data exchange between authenticated ITS stations, all senders of a message in the network (inside the sender's communication range in case of single-hop and additionally outside its range in case of multi-hop message transmission) have to be attested by a central authority. This central authority is shown in Figure 1.1 as Public Key Infrastructure consisting of RCA, LTCA, and PCA. The PKI concept described in the following is consistent with the PKI proposal of the C2C-CC [2] where PRESERVE partners were involved in its creation in lead positions. It is also compatible with the ETSI PKI architecture [14].

All ITS stations (i.e. vehicles, roadside stations, central stations) participating in the ITS G5A communication must be equipped with a VSS that stores valid certificates issued by a trusted Pseudonym Certificate Authority (PCA). The following steps are necessary to actively participate in the network.

1. In a registration process the VSS of a station is equipped with an Long-Term Certificate (LTC) that is issued by a Long-Term Certificate Authority (LTCA).
2. The private key belonging to the LTC is subsequently used to sign a request for Pseudonym Certificates (PCs) containing public keys. The request is sent to a PCA.

3. After the PCA has verified the validity of the LTC the PCA issues the pseudonym certificates using the provided public keys and its own private key. When the PCs are created the PCA sends them to the respective ITS station. In order to prevent a linkablity between the PCs and the LTC, the verification of the LTC is performed by the LTCA.
4. The ITS station can use a PC to sign or encrypt a message in the ITS-G5A communication. As explained in Section 2.4.4, it is not allowed for privacy reasons to use the long-term certificate to sign messages in V2X communications.

Based on these cryptographic mechanisms the most important security goals can be achieved – the exclusion of external attackers. Internal attackers however possess valid credentials and necessary communication technology to overcome these proactive security mechanisms. To reduce the risk of having internal attackers the systems of the communication endpoints should additionally be protected by firewalls and trusted computing solutions [19, 28]. Manipulation of vehicular systems should be made difficult to impede side channel attacks [37] and unintended software manipulation (e.g. flashing of system software [27] or exploiting vulnerabilities). The VSA is aiming for securing the complete internal on-board network of ITS stations to seamlessly protect data on its way from the source of information such as a sensor to the destination such as a display or transmitter. For example, in the use case *Emergency Electronic Brake Lights* [4], information from a braking vehicle has to be transmitted to neighboring vehicles whereupon the sender has to secure every component, interface, and network between the brake sensor and the transceiver. Additionally, the receiving vehicle has to secure every component, interface, and network between the transceiver and the Human Machine Interface (HMI) in order to be sure that an attacker has not manipulated the information [19]. Full protection of data on the way between the source component of the sender (e.g. braking sensor) and destination component at the receiver (e.g. display) is a complex task. However, even if all channels are fully protected by means of cryptography the physical manipulation of sensor inputs cannot be prevented. An attacker could for example manipulate the Global Navigation Satellite System (GNSS) signal that is received and processed by the ITS stations. Consequently, detecting malicious behavior of internal attackers is an additional task that is considered within the VSA.

## 1.3 V2X Privacy Concepts

In addition to the cryptographic mechanisms that take care of sender authentication, message integrity, and optionally for data confidentiality, the privacy of the driver has to be protected. That means, a receiver of V2X messages must not be able to track and identify other stations over long periods of time by monitoring the wireless channel. Since the stations can be identified by several IDs and static data elements contained in V2X packets, a frequent and simultaneous change of all identifiers is necessary. Signer information in the security message header contains also identifying information such as the sender's certificate. As a result several unlinkable pseudonym certificates are used by the stations.

## 1.4 Document Structure

This document is structured as follows. After providing an overview of all relevant ITS security elements of the VSA in this chapter, the on-board security architecture is described in Chapter 2. Based on an abstract on-board security architecture described in Section 2.1 a PRESERVE specific architecture is presented in Section 2.2. Subsequently, the security aspects related to the abstract on-board security architecture are discussed in Sections 2.3 to 2.9. The mechanisms discussed in these sections are primarily based on results of the SeVeCom, the EVITA, and the PRECIOSA project. The security infrastructure in form of a PKI is presented in Chapter 3. Chapter 4 finally concludes this deliverable.

# 2 On-board security architecture

In this chapter the security elements of the security architecture are discussed that are placed inside a vehicle or a roadside station. Most elements of this PRESERVE VSA are based on security solutions from the previous projects SeVeCom, EVITA, and PRECIOSA. The term *on-board* is used in the following to describe all systems and networks that are placed inside an ITS station.

The main focus of the on-board security subsystem is to enable secure communication with other ITS stations via ITS-G5A and to protect the own on-board communication by allowing interoperability with EVITA components in other parts of the vehicle.

In Section 2.1 an abstract on-board security architecture is proposed. This abstract architecture is subsequently used in Section 2.2 to describe the PRESERVE on-board security architecture. In the subsequent Sections 2.3, 2.4, 2.5, 2.6, 2.7, 2.8, and 2.9 details regarding the different on-board VSA elements are provided.

## 2.1 Abstract On-board Security Architecture

The abstract security architecture depicted in Figure 2.1 is based on the European ITS reference architecture described in ETSI EN 302 665 [5]. In our abstract architecture the security layer of the ETSI reference architecture is used and extended by components that are required for a comprehensive ITS-S security solution. Additionally, our abstract architecture allows a simple structuring of the PRESEVE VSS detailed in Section 2.2. The security layer is designed in our architecture as a vertical layer providing security services to the different layers of the communication stack located on the left hand side and the applications located above, cf. Figure 2.1. A detailed description of the interfaces and the connected layers is given in Section 2.9.

The security layer functionality is divided in our abstract architecture into the six components: *secure information*, *secure communication*, *security management*, *security analysis*, *security policies*, and *cryptographic operations*. The tasks and functionalities of these components are described in the following.

**Secure Communication** The secure communication subsystem is strongly related to the topic of secure information. As shown in Figure 2.1 we focus in this document on internal and external communication security. For internal communication data such as sensor measurements, commands or signals have to be transmitted in a secure way for

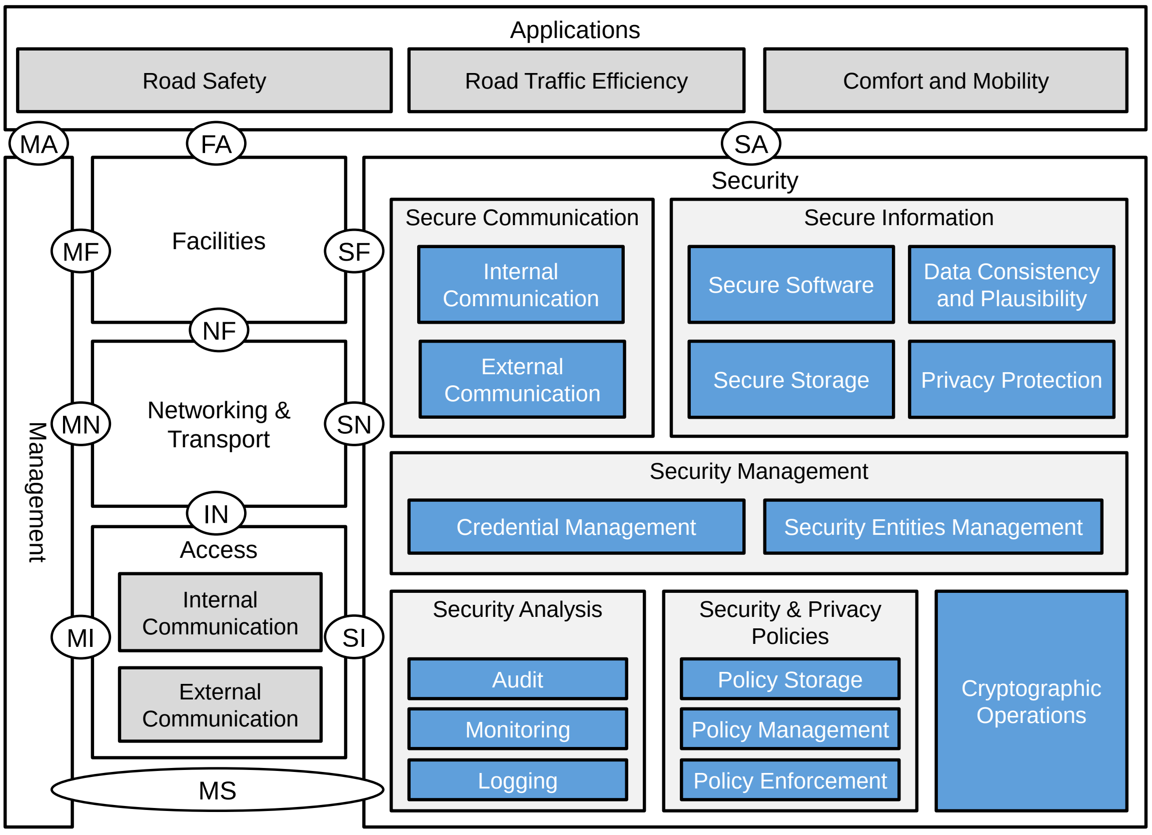


Figure 2.1: Abstract PRESERVE vehicle security architecture

example from a sensor to a data processing control unit (e.g. ECU) in order to prevent manipulation. For external communication the receiver has to verify at least the authenticity and authorization of the sender as well as the integrity of the transmitted data.

The secure communication subsystem is a central component of the abstract architecture that is related to most other security components depicted in Figure 2.1. In order to ensure for example the authenticity of end points in the internal and external communication, the security entities management is used and for checking the sender's authorization the security policies are used. The communication interfaces are further monitored by components of the security analysis in order to detect misuse or attacks. Finally, the cryptographic operations are involved together with the credential management to ensure the integrity and confidentiality of transmitted data.

Details about the secure communication components are provided in Section 2.8.

**Secure Information** The secure information subsystem deals with the protection of all information stored in or exchanged by the ITS station. This includes secure storage, secure software, privacy protection, and data consistency and plausibility. Since V2X applications might be related to the vehicle's safety the consumed data such as on-board sensor information and received V2X message data must be trusted. As a consequence the data has to be gathered, processed, and stored in a secure way. An attacker must not be able to insert new fake information or alter existing information stored inside the ITS-S. Additionally, it has to be ensured that V2X related information is only processed by trustworthy software. For example, an attacker must not be able to install malware that is able to misuse the components of the VSS such as the cryptographic operations to sign fake information. Even if the information is securely gathered and correctly processed the semantics of the information should be verified by data consistency and plausibility checks. Especially if information is received from external parties the correctness of the message content should be verified. Inconsistent or implausible information from internal attackers should be detected to avoid their processing by local V2X applications and prevent false driver notifications or warnings.

In addition to the protection of the data the protection of the drivers' privacy is important. Vehicles must not reveal private information. Moreover, privacy-related information such as location-based data should be anonymized to prevent the identification of individuals based on statistics. The privacy protection subcomponent should ensure that identifying information is removed from sensitive data and privacy-related data is made inaccurate.

The related security components of the specific PRESERVE architecture that are related to secure information are further discussed in Section 2.4.

**Security Management** The security management is responsible for the organization of credentials that are required by the secure communication. The credential management provides the access to credentials of the local ITS-S such as keys or certificates. However, especially private keys as sensitive information must not be stored in local databases or files. By using the secure storage subcomponent sensitive data is stored in a HSM to prevent extraction or misuse. Other credentials such as the root certificate(s) of the PKI must be stored in a secure way so it cannot be substituted by attackers. Public key material related to internal entities (e.g. local sensors or ECUs) and external entities (e.g. V2X neighbors) is managed by the security entities management subcomponent. By using the security policies and the security analysis the entities management can control the access from and to entities and might be able to reconfigure a firewall module with new filtering rules. For example, only the security subsystem might be allowed to establish a secure connection between entities of the local on-board network. Additionally, the entities management organizes identities and trust levels of external ITS-S neighbors.

The related security components of the specific PRESERVE architecture that are related to security management are discussed in detail in Section 2.5.

**Security Analysis** The security analysis includes the tasks of monitoring, auditing, and logging of information that might be relevant for the VSS. The main objective of the security analysis component is the verification of the security system status and to evaluate results of the security policies. Based on an Intrusion Detection System (IDS) it allows the detection of security risks or attacks. The IDS might detect not only intrusions to the ITS-S through external V2X communications but also intrusions to internal vehicle networks performed by attackers with physical access. Moreover, to protect the ITS station from unauthorized access based on filter rules, a firewall might also be required. Together with the data consistency and plausibility checks being part of the secure information subcomponent the security analysis might be able to detect abnormal or faulty behavior of the local system. The security audit subcomponent could then analyze collected information (e.g. logging data) and react by updating policy rules.

The security components of the specific PRESERVE architecture that are related to the security analysis are further discussed in Section 2.7.

**Security & Privacy Policies** Security and privacy policies are managed, stored and enforced by the security and privacy policies subsystem. It defines for example authorization access control list rules specifying which applications or components of ITS-S are authorized to access system resources. Likewise, this component can also manage policies related to pseudonym management and other aspects of privacy protection. Based on results of the security analysis and logging data, the security policies could be updated.

More information about related PRESERVE components are provided in Section 2.6.

**Cryptographic Operations** The cryptographic operation subsystem provides basic security functions like encryption, decryption, signature generation and verification. At least these operations are required to perform secure internal and external communications. If the cryptographic operations are implemented within an HSM additional functionality could be provided such as tamper resistant storage, a secure time base and random number generation that can be used by other components of the security layer. Moreover, with an HSM sensitive cryptographic functions can be performed with keys stored inside the HSM, for example encryption, decryption, and signing. Even if we consider that a dedicated hardware module is needed to meet security requirements of secure V2X communication a software library could alternatively or in addition be used to perform specific cryptographic operations.

Security components of the PRESERVE architecture that are related to cryptographic operations are further discussed in Section 2.3.

## 2.2 PRESERVE On-board Security Architecture

Based on the abstract security architecture described in Section 2.1, a more specific PRESERVE architecture is discussed in this section. This specific architecture aims to reuse logic components and available implementations of the previous projects SeVeCom, EVITA, and PRECIOSA. The relation between the abstract security architecture and the PRESERVE architecture is discussed in Section 2.2.1. An overview of the PRESERVE vehicle security architecture is presented in Figure 2.2. On the left hand side of this Figure, the external on-board elements are displayed that are connected with the VSS that is visible as blue box on the right hand side.

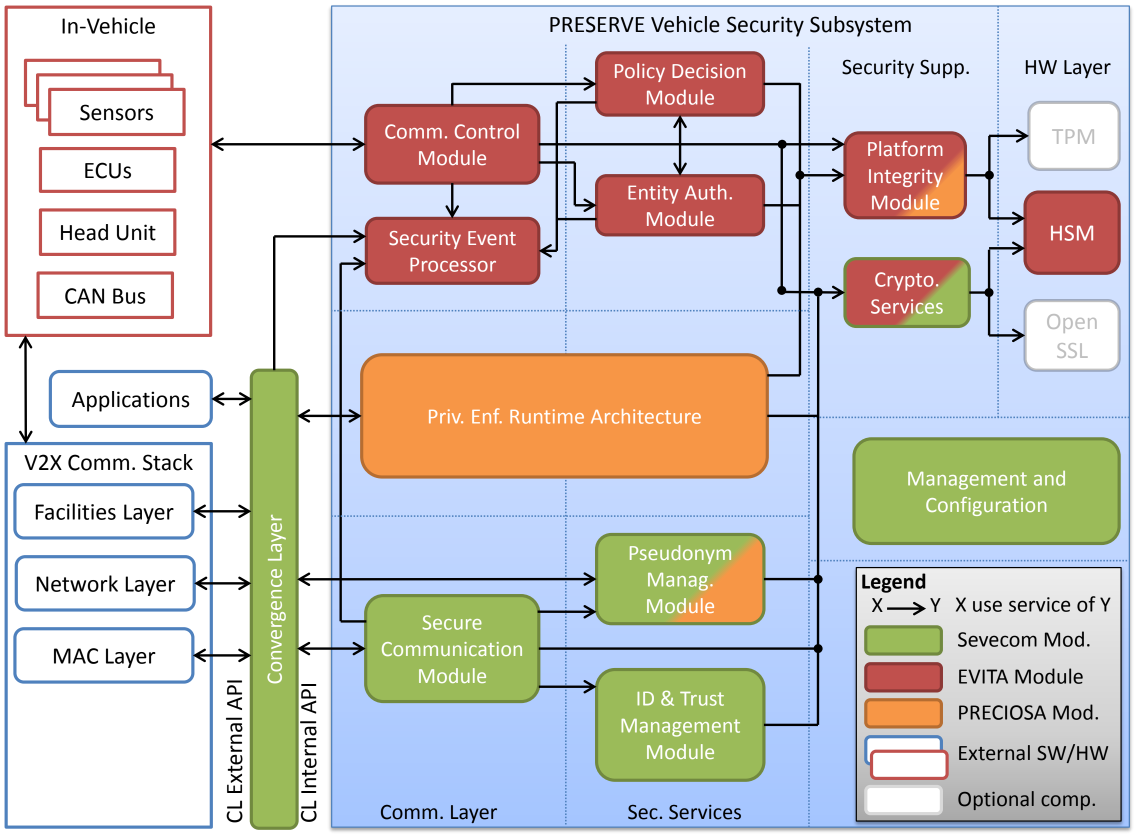


Figure 2.2: PRESERVE vehicle security architecture overview

The external elements on the left hand side of Figure 2.2 can further be divided into controlled and uncontrolled parts.

1. The on-board elements such as sensors, ECUs, head unit or CAN bus may be controlled by the VSS as long as they are equipped with their own HSM following EVITA specifications [40]. A more detailed description regarding controlled external on-board elements can be found in Section 2.8.2.

2. The V2X communication stack instead is not controlled by the VSS. This external element is connected with the VSS in order to use its security services to protect messages transmitted in the ITS-G5A communication. On the one hand, data is exchanged directly over an API. This API is provided in PRESERVE by the convergence layer (CL) that is presented in Section 2.9.1.3. On the other hand, data is exchanged indirectly through the communication stack in order to transport security related information that is associated to a specific message or a V2X network neighbor. This type of information is called *meta data* and is used to append information to a message that is moved through the V2X communication stack (e.g. security processing requirements or security processing results). Details regarding this meta data concept can be found in Section 2.9.2.

Internal VSA elements shown on the right hand side of Figure 2.2 can be divided into three groups. The green elements at the bottom of this figure are responsible for V2X communication message protection and base mainly on results from the SeVeCom project [25]. In order to have a flexible vehicular security architecture that can be used in different current and future FOTs, the dependencies between the VSS software groups should be kept as minimal as possible. This means, that the components of the V2X communication subsystem must not require functionalities of the on-board communication subsystem or the Privacy-enforcing Runtime Architecture (PeRA) to be operational. However, all software components of the PRESERVE VSS depend on cryptographic operations of the cryptographic services module to access the HSM as shown on the right hand side of Figure 2.2. The cryptographic services module is flexible regarding the usage of different cryptographic providers (i.e. software library or hardware security module). If a software library is used, an optional Trusted Platform Module (TPM) can be used as secure key storage and as root of trust. If an HSM is available, the hardware interface to this module is used instead of the software library. The HSM provides access to the secured storage and HW accelerated cryptographic operations. Both, secured storage and crypto acceleration are necessary to process a large number of data packets in a secure manner.

The red elements, shown at the top of Figure 2.2, are responsible for on-board system security and on-board communication protection. These elements are reused from the EVITA [40] project.

The orange elements in the middle of Figure 2.2 provide mechanisms for privacy protection of the ITS station and respectively its driver or owner. Considering a FOT that does not implement such applications the integration of the Privacy-enforcing Runtime Architecture (PeRA) inside the VSS is not necessary.

### 2.2.1 Relation Between the Abstract Architecture and the PRESERVE Architecture

In Table 2.1 a mapping of the components of the PRESERVE architecture with the components of the abstract architecture is presented. Since the PRESERVE components are based on previous project results the functionality of the PRESERVE components is related in some cases to several logical components of the abstract architecture.

Table 2.1: Mapping of PRESERVE architecture components with logical components of the abstract architecture

| | PRESERVE Architecture | Abstract Architecture |
|---|---|---|
| Sevecom | Secure Communication Module (SCM) | Secure Communication / External Communication (Section 2.8.1) |
| | | Interfaces of the On-Board V2X Security Subsystem (Section 2.9) |
| | Pseudonym Management Module (PMM) | Privacy Protection (Section 2.4.4) |
| | | Credential Management (Section 2.5.1) |
| | | Security Policies (Section 2.6) / Policy Enforcement |
| | Identification and Trust Management Module (IDM) | Credential Management (Section 2.5.1) |
| | Convergence Layer | Interfaces of the On-Board V2X Security Subsystem (Section 2.9) |
| | Management and Configuration | Security Management (Section 2.5) |
| EVITA | Communication Control Module (CCM) | Secure Communication / Internal Communication (Section 2.8.2) |
| | Policy Decision Module (PDM) | Security Policies (Section 2.6) / Policy Management and Policy Storage |
| | Platform Integrity Module (PIM) | Secure Information / Secure Software (Section 2.4.2) |
| | Cryptographic Services (CRS) | Cryptographic Operations (Section 2.3) |
| | | Secure Storage (Section 2.4.1) |
| | | Credential Management (Section 2.5.1) |
| | Security Event Processor (SEP) | Security Analysis (Section 2.7) |
| | | Data Consistency and Plausibility (Section 2.4.3) |
| | | Security Policies (Section 2.6) |
| PRECIOSA | Privacy-enforcing Runtime Architecture (PeRA) | Privacy Protection (Section 2.4.4) |

### 2.2.2 V2X Security Subsystem as Part of the On-board Architecture

The PRESERVE V2X Security Subsystem (VSS) aims to be flexibly usable in different field operational tests for securing external V2X communications and internal on-board communications. Therefore, the external interfaces of the VSS are well defined in order to use them appropriately for different integrations. The following connection points are required by the PRESERVE VSA to connect with external elements of an on-board ITS station.

- **Convergence layer API**
  The convergence layer is the main interface between the V2X communication stack and the VSS. As defined in the ETSI ITS station reference architecture [5, Section 4.4] and depicted in Figure 2.1 there are five interfaces between the security subsystem and the communication stack.
  - MS (Management $\leftrightarrow$ Security)
  - SI (Access $\leftrightarrow$ Security)
  - SN (Network & Transport $\leftrightarrow$ Security)
  - SF (Facilities $\leftrightarrow$ Security)
  - SA (Applications $\leftrightarrow$ Security)

  The API of the convergence layer provides interfaces to the different layers for using security services. The proposed access methodology presented in Section 2.9.1.3 aims at being compliant with ongoing standardizations efforts in ETSI as described in Section 2.9.1.
- **Meta data exchange**
  The exchange of message related meta data, described in Section 2.9.2, is used to transmit security information between its generator (e.g. VSS) and processor (e.g. application). In case of message distribution, the application or the facilities layer generates a message, adds security processing meta data to it and gives both to the communication stack. When the VSS receives the data subsequently on the respective layer, the message can be processed according to the requirements stated in the meta data. In case of processing a received message the VSS appends the result of the security operation as meta data to the message and submits it to the communication stack. The application finally can use message or station related meta data in order to react accordingly.
- **Secure on-board communication**
  The on-board communication is protected by secure internal communication modules discussed in Section 2.8.2. The modules are directly connected with other on-board devices such as sensors or ECUs that are equipped with an EVITA HSM.
- **Access to local on-board data**
  In order to check the content of incoming V2X messages the VSS needs access to the on-board network (e.g. CAN bus, GNSS receiver, local sensors) to retrieve status information about the own stations. For data consistency and plausibility checks it is necessary to get frequently updated station position information, its speed and heading, a GNSS synchronized time base and possibly environment sensor data such as radar.

## 2.3 Cryptographic Operations

This section describes the cryptographic support subsystem, which is used by most components of the PRESERVE VSS. The components of the PRESERVE VSS can use the Cryptographic Services (CRS) module that provides a well-defined homogeneous interface for performing all kinds of cryptographic operations and handling security credentials such as keys. The actual implementation of the cryptographic operations is done in software crypto library or in a Hardware Security Module (HSM). In addition, the CRS module can also use a software library like OpenSSL as a backup component for algorithms that are not implemented in the HSM. However, accessing the HSM is the preferred way for performing cryptographic operations since it offers hardware acceleration and secure storage.

### 2.3.1 Cryptographic Services

The Cryptographic Services module provides cryptographic services such as encryption/decryption, signature generation/verification and creation of random numbers to other components of the PRESERVE VSS. The CRS module is an abstraction layer that hides implementation details of the underlying cryptographic operations. By default, the CRS module performs cryptographic operations by calling the corresponding functions of the HSM interface as shown in the sequence diagrams for message processing in Sections 2.8.1.1 and 2.8.1.2. However, if a cryptographic algorithm is not available on the HSM a software cryptographic library like OpenSSL is used instead. This is completely transparent for the modules that are using the CRS module. The modular concept was already proposed both in the EVITA project [40] and in SeVeCom.

**Usage of the Cryptographic Services** As mentioned before, the cryptographic services are used by almost all components of the PRESERVE VSS. Some exemplary usages of the CRS by other modules are listed in the following.

- The Secure Communication Module uses the cryptographic services in order to sign outgoing messages such as CAMs and DENMs. For incoming V2X messages the verification service is used to verify the authenticity of the sender and the integrity of the message content.
- The verification of the certificate of an external V2X neighbor is the task of the Secure Communication Module which in turn uses the CRS.
- When encryption is requested, the Secure Communication Module uses the CRS to encrypt or decrypt V2X messages.
- When the Pseudonym Management Module needs more pseudonyms (e.g. when the number of available pseudonyms is below a threshold) it calls the ID & Trust Management Module (IDM). The IDM uses the cryptographic services to create new key pairs that are used in the request of new pseudonym certificates. In this process

the private keys remain in the HSM and are stored in a secured way within the secure storage (TPM). In order to have a link between the pseudonym certificate, the public key, and the appropriate private key the CRS module provides a unique key handle ID for this purpose.

- The Communication Control Module (CCM) uses the cryptographic services of the CRS for encrypting/decrypting and signing/verifying messages sent over a secure channel controlled by the CCM.
- The Entity Authentication Module (EAM) uses the CRS for signing and verifying authentication tickets applied in internal on-board communications.
- The Platform Integrity Module (PIM) uses the CRS for performing encryption on updating the measurement chain and for performing decryption on delivering elements of the measurement chain. Furthermore, it uses the CRS to sign trust statements.

**Timing**

- The CRS is a proxy for the explicit cryptographic service (i.e. sign, verify, encrypt, decrypt, key generation) performed by libraries or cryptographic devices. Therefore, the latency introduced by this component must be kept as low as possible.
- The CRS is used for every incoming and outgoing message by the different components. Therefore, low latency requirements are requested as defined in the requirements analysis of the PRESERVE deliverable D1.1 [36, Section 3.1.8].

**Relevant Interfaces to Modules of the PRESERVE Architecture** The CRS module comprises the following functions.

- Key import and export functions
- Certificate verification and public key extraction functions
- Key creation, derivation, and validation functions
- Functions that encrypt and decrypt data
- Functions that sign data and verify the signature of data
- Random number generation functions
- Hash creation functions
- Signature and MAC creation and verification functions

The functions of the CRS module can be distributed into four groups. The first one comprises the lower level functions listed in Table 2.2 that manage only cryptographic concepts. The second one comprises the high level functions listed in Table 2.3 that, in addition to the previous ones, are able to manage concepts related to the common constructions to both standards considered by PRESERVE (IEEE 1609.2 and ETSI TS 103 097). The third groups contains the functions that are currently only used by the CRS module (i.e. internal functions). In Table 2.4 the relevant internal functions are listed. The last group contains the functions as listed in Table 2.5 that allow the management of missing certificates. The later functions are primarily used by the IDM, cf. Section 2.5.1.1.

Table 2.2: Low level functions of CRS

| Name | Description |
|---|---|
| CRS_CalculateHash | Calculates a hash of data |
| CRS_CcmCipher | Low level function that ciphers data |
| CRS_CcmDecipher | Low level function that deciphers data |
| CRS_CreatePrivateKey | Generates a key pair for encryption/decryption or signing/verifying |
| CRS_CreateRandom | Generates a random number |
| CRS_ExportEciesKey | Exports a symmetric key encrypted using ECIES encryption scheme. The exported key data can be used directly to build a certificate request |
| CRS_ImportEciesKey | Imports an encrypted symmetric key using ECIES encryption scheme |
| CRS_ReleaseKey | Removes a key (private, public or symmetric) |
| CRS_Sign | Signs data |
| CRS_Verify | Verifies the signature associated to a data |

Table 2.3: High level functions of CRS

| Name | Description |
|---|---|
| CRS_Cipher | Encrypts a data using the function CRS_CreateAesCcmCiphertext and put the result in an IEEE or ETSI object |
| CRS_ComputeCertId8 | Creates a digest from a data (with the 8 least significant bytes of the hash of the data) |
| CRS_CreateETSIPublicKey | Creates a public key for the standard ETSI TS 103 097 |
| CRS_CreateIEEEPublicKey | Creates a public key for the standard IEEE1609.2 |
| CRS_CreateRecipientInfo | Creates a RecipientInfo object which contains information on the keys that have been used for the encryption of the data. It calls the function CRS_CreateEciesNistP256EncryptedKey |
| CRS_Decipher | Decrypts an encrypted message |
| CRS_GetSignature | Returns the low level part of a signature |

| Name | Description |
|---|---|
| CRS_GetStoredKey | Retrieves the public key associated to a certificate request |
| CRS_StoreKeyForCertificate | Stores a public key generated during the creation of a certificate request (long term certificate or pseudonym certificates). This process has two purposes. 1) to retrieve the key pair associated to a certificate after a positive response of the CA. 2) to remove the key pair in case of an error. |
| CRS_StorePublicKey | Imports a public key (see function CRS_ImportPlainKey below) and stores it temporarilly |
| CRS_SignCertificate | Signs a certificate. This function is only for test purposes. |
| CRS_SignMessage | Signs a message that must be sent |
| CRS_VerifyCertificate | Verify the signature of a certificate |
| CRS_VerifyMessage | Verify the signature of a message |
| CRS_VerifySignature | Verify the signature of a general component such as an array of bytes |

Table 2.4: Internal functions of CRS

| Name | Description |
|---|---|
| CRS_CalculateHashOfPublicKey | Calculates the hash of public key data |
| CRS_ChangeEndianness | Modifies the endianness of data before passing it to the cryptographic layer |
| CRS_CreateAesCcmCiphertext | Creates a symmetric key and encrypts this key. It returns an object of type AesCcmCiphertext according to IEEE 1609.2 and ETSI TS 103 097 |
| CRS_CreateEccPoint | Creates a key pair and put its public part in an EccPoint (ETSI object) |
| CRS_CreateEccPublicKey | Creates a key pair and puts its public part in an EccPublicKey object according to IEEE 1609.2 |
| CRS_CreateEciesNistP256EncryptedKey | Exports a public key using the function CRS_ExportEciesKey and puts the result in an object of type EciesNistP256EncryptedKey according to IEEE 1609.2 and ETSI TS 103 097 |
| CRS_GetPublicKey | Returns the public part of a key pair |
| CRS_ImportEciesNistP256EncryptedKey | Imports a public key using the function CRS_ImportEciesKey from an object of type EciesNistP256EncryptedKey |
| CRS_ImportPlainKey | Imports a public key from a certificate |

Incoming signed messages can contain either the complete certificate of the signer or only the digest of this certificate. In the latter case, it may happen that the corresponding certificate is not known by the VSS. However, if a complete certificate is provided, it may happen during the verification of the certificate chain that the certificate of the CA used to sign the sender's certificate is not known by the VSS. When the CRS module detects one of these situations it calls the IDM module which takes the appropriate decision.

Table 2.5 shows functions of the CRS module that manage theses situations.

Table 2.5: Management of missing certificates using the CRS

| **Name** | **Description** |
|---|---|
| CRS_GetMissingATCert | Returns the certificate ID of the missing PC |
| CRS_GetMissingAACert | Returns the certificate ID of the missing CA certificate |
| CRS_SetMissingATCert | Saves the certificate ID of the missing PC |
| CRS_SetMissingAACert | Saves the certificate ID of the missing CA certificate |
| CRS_ResetMissingATCert | Resets the certificate ID of the missing pseudonym certificate when the related information has been used |
| CRS_ResetMissingAACert | Resets the certificate ID of the missing CA certificate |

### 2.3.2 OpenSSL

When the cryptographic primitives of the HSM are not available, the CRS uses the cryptographic services of the OpenSSL library. For PRESERVE, OpenSSL can be viewed as a backup between VSS modules and the HSM. OpenSSL is an open source implementation of the two protocols SSL (Secure Sockets Layer) and TLS (Transport Layer Security). Furthermore, OpenSSL provides a library of all sorts of cryptographic primitives with very efficient implementations. OpenSSL is delivered as a toolkit which provides a lot of commands and as a library which can be used by applications like the VSS. It offers a lot of functionalities among which only the following are used by the PRESERVE VSS. These allow the CRS module to have the same behavior whatever underlying cryptographic layer applies.

- Key import and export functions
- Certificate verification and public key extraction functions
- Key creation, derivation and validation functions
- Symmetric encryption/decryption (e.g. blowfish, cast, des, idea, rc2, etc)
- Asymmetric encryption/decryption (e.g. dsa, rsa)
- Signature of data and verification of the signature of data
- Random number generation functions
- Cryptographic hash functions (e.g. md2, md4, md5)

- Message authentication code (e.g. hmac)

The CRS module uses the functions listed in Table 2.6. More functions are certainly used internally but not directly called by the CRS module.

Table 2.6: Functions of OpenSSL used by the CRS module

| Name | Description |
|---|---|
| AES_set_encrypt_key | Initializes an AES key |
| BN_bn2bin | Converts a big number (of type BIGNUM) into an array of integers. |
| BN_clear_free | Releases the memory associated to a big number |
| CRYPTO_ccm128_encrypt | Realizes the encryption process |
| CRYPTO_ccm128_init | Prepares the internal data for the encryption process |
| CRYPTO_ccm128_setiv | Sets-ups the nonce and the length of the message to encrypt |
| CRYPTO_ccm128_tag | Returns the tag calculated during the encryption process |
| ECDH_compute_key | Computes the shared secret using the ECDHC method |
| ECDSA_do_sign | Signs a data element |
| ECDSA_do_verify | Verifies the signature associated to a data element |
| ECDSA_SIG_new | Allocates memory for a signature |
| EC_GROUP_free | Releases the memory associated to a elliptic curve group |
| EC_GROUP_new_by_curve_name | Creates the elliptic curve groups |
| EC_GROUP_precompute_mult | Does the pre-computation during the initialization phase in order to improve the performance |
| EC_KEY_free | Releases the memory associated to an elliptic curve key |
| EC_KEY_generate_key | Creates a new elliptic curve private (and optional a new public) key |
| EC_KEY_get0_public_key | Returns the elliptic curve group associated to an elliptic curve key |
| EC_KEY_new | Allocates the memory for an elliptic curve key |
| EC_KEY_set_group | Sets the elliptic curve group of a elliptic curve key object. |
| EC_KEY_set_public_key | Sets the public key of a elliptic curve key object |
| EC_POINT_free | Releases the memory associated to an elliptic curve point |
| EC_POINT_get_affine_coordinates_GFp | Gets the affine coordinates of an elliptic curve point over Galois field GFp |
| EC_POINT_new | Allocates the memory for an elliptic curve point |

| Name | Description |
|---|---|
| EC_POINT_set_affine_coordinates_GFp | Sets the affine coordinates of an elliptic curve point over Galois field GFp |
| EC_POINT_set_compressed_coordinates_GFp | Sets the x9.62 compressed coordinates of a elliptic curve point over Galois field GFp |
| EVP_DigestFinal | Retrieves the digest value from the default digest context |
| EVP_DigestInit | Sets up the digest context to the default digest implementation |
| EVP_DigestUpdate | Hashes data into the default digest context |
| EVP_MD_CTX_init | Initializes the digest context passed to it |
| EVP_MD_CTX_create | Creates a digest context |
| EVP_sha256 | Implement the digest sha256 algorithm |
| HMAC | Computes the HMAC of the ciphered data. It is used for importing or exporting keys |
| OpenSSL_add_all_algorithms | Initializes the OpenSSL specific *EVP* algorithm |
| OpenSSL_add_all_digests | Initializes the OpenSSL specific *EVP* hash |
| RAND_pseudo_bytes | Generates random data |

### 2.3.3 Hardware Security Module

The Hardware Security Module (HSM) is a cryptography coprocessor to perform cryptographic operations like encryption/decryption, signature generation/verification, creation of random numbers and the secure storage of keys. It is based on the HSM used in EVITA [40] that already provides most of the desired functionality. However, some extensions in the cryptographic functionality as well as modifications in the interface have to be made to this HSM to fulfill the needs of PRESERVE. The PRESERVE HSM supports the cryptographic algorithms listed in Table 2.7.

Table 2.7: Functions of HSM as described in EVITA deliverable D3.2 [40]

| Crypto primitive | Mode of operation | Key length | Requirement origin |
|---|---|---|---|
| AES | CTR | 128 | [22, Section 5.2.22 and 6.4.3.1.6], basis for CCM mode |
| AES | CBC_MAC | 128 | [22, Section 5.2.22 and 6.4.3.1.6], basis for CCM mode |
| AES | GCM | 128 | [40, Section 4.2.3.1] |
| AES | OMAC1 (CMAC) | 128 | |
| AES | CCM | 128 | [40, Section 4.2.3.1] |
| AES | PRNG | 128 | [40, Section 4.2.3.1] |

| Crypto primitive | Mode of operation | Key length | Requirement origin |
|---|---|---|---|
| ECC | NIST P224 | | [22, Section 5.2.16] (added for PRESERVE) |
| ECC | NIST P256 | | [22, Section 5.2.16] |
| ECIES | NIST P256 | | [22, Section 5.2.16] (added for PRESERVE) |
| SHA-256 | | | [22, Section 5.3.32] |
| WHIRLPOOL | | | [40] (4.2.3.1), implemented in EVITA as an exemplary SHA-3 candidate |
| HMAC | HMAC-SHA1 | | [40] (4.3.6.3) |
| Monotonic counter | | | [40] (4.2.3.1) |
| Secure clock | | | [40] (4.2.3.1) |
| Key generation | AES | 128 | |
| Key generation | ECC | 224/256 | |
| Key generation | ECIES | 256 | added for PRESERVE |
| TRNG | | | added for PRESERVE |

The CRS modules uses the function listed in Table 2.8.

Table 2.8: Functions of the HSM used by the CRS module

| Name | Description |
|---|---|
| HSM_Cipher_Finish | Finishes the cipher process |
| HSM_Cipher_Init | Begins the cipher process |
| HSM_Cipher_Update | Realizes the cipher process |
| HSM_Create_Random_Key | Creates a random key |
| HSM_Decipher_Finish | Finishes the decipher process |
| HSM_Decipher_Init | Begins the decipher process |
| HSM_Decipher_Update | Realizes the decipher process |
| HSM_Export_ECIES_Enc_Key | Exports the ECIES encrypted key data of a symmetric key |
| HSM_Get_Random | Generates a random number |
| HSM_Hash_Finish | Finishes the sign process |
| HSM_Hash_Init | Begins the sign process |
| HSM_Hash_Update | Realizes the sign process |
| HSM_Import_ECIES_Enc_Key | Imports the ECIES encrypted key data of a symmetric key |
| HSM_Import_Plain_Key | Imports a plain key |
| HSM_Key_Remove | Removes a key (private, public or symmetric) |
| HSM_Key_Status | Returns some information on a key |
| HSM_Sign_Finish | Finishes the sign process |
| HSM_Sign_Init | Begins the sign process |
| HSM_Sign_Update | Realizes the sign process |

| Name | Description |
|---|---|
| HSM_Verify_Finish | Finishes the verify process |
| HSM_Verify_Init | Begins the verify process |
| HSM_Verify_Update | Realizes the verify process |
| HSM_Verify_Messages | This is an alternative to the three functions above |

Each function of the HSM returns an error code which indicates either a successful completion of the command or in case of an error it returns the error category. A listing of possible error codes is omitted at this place because it might be subject to change during the implementation. However, an incomplete description of possible error codes can be found in the EVITA deliverable D3.2 [40, Section 4.3.3] and in the function descriptions of [40, Section 4.3.5].

The basic cryptographic operations that process input data inside the HSM are split into three sub operations.

- Init for setting up a session and passing initial parameters to the HSM
- Update for processing a block of data. In this step the to be processed data is handed over. The result of the operation is provided in this step or in the subsequent finish step.
- Finish for performing possible finishing operations and destroying the session

Splitting the cryptographic operations into init, update, and finish simplifies the handling of the data in the HSM and furthermore provides the possibility of performing multiple cryptographic operations in parallel by using sessions for each operation. Since the frequency of message verification may be very high (possibly 1000 message verifications per second according to the PRESERVE deliverable D1.1 [36], Section 3.1.8), the function *HSM_Verify_Messages* performing verification of several messages in one step is introduced to decrease the overall latency of message verification. A detailed description of the supported functions can be found in the EVITA deliverable D3.2 [40, Section 4.3.5].

## 2.4 Secure Information

Secure information clusters several important functionalities of the V2X on-board security architecture as illustrated in Figure 2.1 on page 8. The secure storage described in Section 2.4.1 ensures that sensitive data is protected accordingly. The secure software described in Section 2.4.2 is responsible to protect against or detect unauthorized system manipulations. Furthermore, data consistency and plausibility checks are relevant in particular for external communications as discussed in Section 2.4.3. Finally, the protection of private and sensitive data is considered in Section 2.4.4.

### 2.4.1 Secure Storage

The secure storage of sensitive data, e.g. private keys, is one of the most important aspects and absolutely necessary in order to achieve an overall secure system architecture. This sensitive data must be prevented to be revealed by an attacker because this would enable him to impersonate as a valid C2X communication partner whose messages will be trusted by other communication partners. Therefore it should not be allowed to store keys and other sensitive data in plain on common non-volatile memory, e.g. flash or $E^2$PROM.

One solution can be to use tamper protected storage devices. These devices have common interfaces for host CPU connection but include, beside the general storage capabilities, multiple sensors that monitor variations of environmental parameters such as voltage, temperature or even light to detect possible hardware attacks. In case an unusual variation is detected these devices are able to erase the storage content immediately or overwrite it with a random pattern. These devices have really good security properties and protect sensitive data very well but the major drawback is their price. Not only the costs for the devices itself, which may be rather high, but a common storage device for program code and data is needed in any case. In the end there are at least two storage devices, one for code and data and one for sensitive data, that must be placed at the Printed Circuit Board (PCB). This may increase the complexity and size of the PCB and therefore its costs. The usage of Trusted Platform Modules (TPMs) for key storage is also possible but unfortunately with the same drawbacks as tamper protected storage devices.

Since the PRESERVE VSS ASIC can support and use tamper protected storage devices and TPMs in general, another usual but much cheaper solution is conceivable for the VSS architecture. Sensitive data can be stored confidential and authentic on a common storage device. Therefore different and unique device keys are used to encrypt a key object (confidentiality) and create a MAC (authenticity) of a stored key object. Due to encryption, an attacker is not able to reveal the key and due to the MAC it is not possible to modify a key object stored in non-volatile memory without noticing. If a modification of a key object is detected its future usage is prevented.

### 2.4.2 Secure Software

The V2X security solution is responsible to prevent the installation of malware and should prevent malicious behavior of installed software in user partitions such as the application unit. The Platform Integrity Module (PIM) is a software component on the PRESERVE VSS that acts as trusted computing base initialization, attestation, and chaining authority. It handles platform initialization by handling the corresponding measurements of the secure boot process, it provides measurements about the system integrity to other autonomous entities and it manages chaining and unchaining of measurements. A more detailed description is given in the EVITA deliverable D3.2 [40, Section 4.4.7] together with an introductory overview of trusted computing in [40, appendix A.2.1.3].

The PRESERVE VSS can additionally be combined with the components of the OVERSEE project [20]. The OVERSEE security architecture basically consists of two parts. The first part is the HSM that provides accelerated cryptographic function execution, secure key and certificate storage, and registers for secure boot services. The second part is a dedicated VM partition for the security services. This partition owns exclusive rights to access the HSM and also hosts building blocks for further security services. The security services partition provides a secure and isolated runtime for the shared security services. Furthermore this partition provides a secure interface for the user partitions to access the security services provided by the secure service partition. This security services partition can host the PRESERVE VSS in a straightforward way, providing additional isolation and thus enhanced security. However, one has to note that message processing would then entail partition changes and thus incur a performance penalty.

The PRESERVE VSS can be integrated with the OVERSEE system in the following way. First, PRESERVE provides a protected environment for storing security sensitive data (cf. Section 2.4.1) and processing V2X messages. Furthermore it features an HSM with HW-accelerated cores for generating and verifying V2X messages. The secure key storage ensures that key data cannot be stolen to generate false V2X messages. However, the system interfacing with the PRESERVE chip can use the key data to generate false V2X messages. As a consequence, PRESERVE and the interfacing system should authenticate each other to ensure that the PRESERVE chip is being used by the correct control unit. Moreover, the PRESERVE VSS could use misbehavior detection mechanisms to prevent signing malicious data, cf. Section 2.7. Another issue is that the software on the control unit could be modified to use the PRESERVE chip to generate false messages. Therefore, secure boot mechanisms can be provided by the PIM module discussed in this section.

The control unit may run different applications (from different suppliers) and furthermore may have a complex operating system (e.g. Linux). This complexity increases the attack surface of the software and most probably brings many vulnerabilities to the architecture. OVERSEE solves this issue by separating the platform into isolated domains. Each domain has its own runtime environment, memory area, CPU resources, and assigned interfaces. With this architecture, software modules can be distributed to these domains according to their criticality. For example third party assistance software running on Java can run on a separate Linux partition, security critical applications (e.g. vehicle warning) can run on a minimal OSEK [1] partition.

Furthermore access to critical hardware interfaces is also separated by the OVERSEE architecture. The actual access to the hardware interfaces and the drivers runs in a separate domain. This has two main advantages:

- No Application or domain has direct access to the hardware.
- Hardware interfaces can be shared among the domains and (fine grained) access policies for each domain can be defined centrally.

[1]Open Systems and their Interfaces for the Electronics in Motor Vehicles

The interfaces are forwarded to the other domains through virtual drivers. A similar approach has been taken for security relevant services. Access to HSM's, certificate services, V2X stacks etc. are running in a separate domain. Access to security services is provided to the other domains by standardized interfaces to a wide extend (PKCS#11, PAM modules, LDAP access etc.). The functionality of PRESERVE could be protected and forwarded in a similar manner.

**Connection to other Modules of the PRESERVE Architecture** As shown in the component diagram of Figure 2.3 the platform integrity module is primarily connected with other EVITA modules. Since the HSM is used as security anchor for handling the platform initialization, it strongly depends on this hardware. In order to access the trusted platform services of the HSM or TPM at least the PeRA module depends on the existence of the PIM.

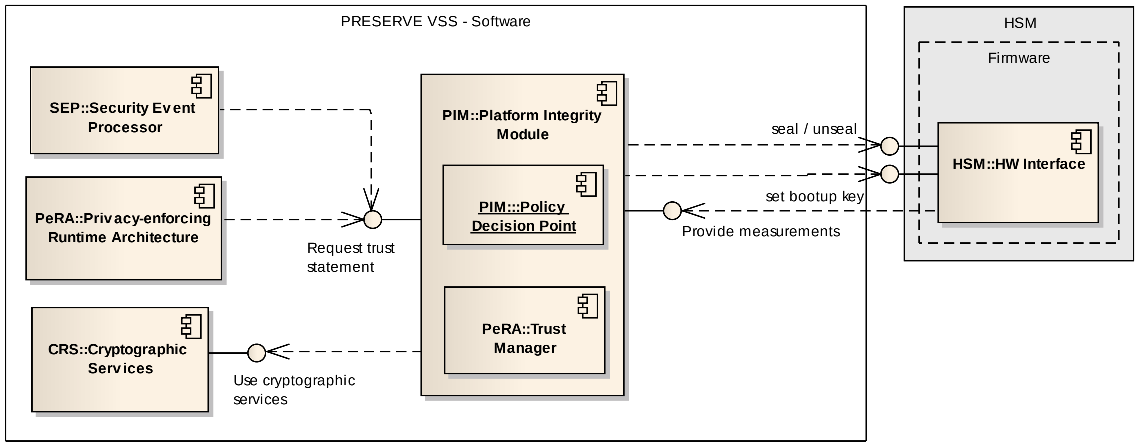

PRESERVE VSS - Software
SEP::Security Event Processor
PeRA::Privacy-enforcing Runtime Architecture
CRS::Cryptographic Services
PIM::Platform Integrity Module
PIM:::Policy Decision Point
PeRA::Trust Manager
Request trust statement
Use cryptographic services
seal / unseal
set bootup key
Provide measurements
HSM
Firmware
HSM::HW Interface

Figure 2.3: Integration of the PIM into the PRESERVE on-board security architecture

**Data Flow** Examples for the data flow between the PIM and the connected modules are shown in Figure 2.3 and briefly explained in the following listing.

- The Security Event Processor (SEP) or the Privacy-enforcing Runtime Architecture (PeRA) access the PIM in order to get information about the VSS integrity. If the vehicular hardware or software is compromised then the SEP and PeRA may reject the usage of security functions.
- The Cryptographic Services (CRS) are used by the PIM to encrypt/decrypt new measurements and to sign/verify trust statements.
- The Hardware Security Module (HSM) is used as security anchor for handling the platform initialization, i.e., the HSM provides initial measurement performed during platform initialization to the PIM.

The following functions provide access to the PIM and can be used to verify the platforms integrity.

- A function that creates a new monotonic counter with the given authentication secret. It returns a counter identifier that acts as handle for referring to the created counter for its usage.
- A function that returns the current counter value of the counter referred by the given counter identifier.
- A function that increments the counter value of the counter referred by the given counter identifier.
- A function that deletes a counter, if the counter exists and the authentication secret is valid.
- A function that extends the measurement chain with a new measurement from a measurement service.
- A function that returns a signed integrity trust statement at the requested position (chain_element_number) from the PIM's internal integrity measurement chain.
- A function that chains (i.e. encrypt) arbitrary data (plaintext) to certain given integrity measurement chain (up_to_chain_element_number) that can be unchained (i.e. decrypted) only under the integrity measurement chain as defined during chaining and optionally only under exactly the same platform if chain_to_platform_identifier is true.
- A function that takes the given chaining_context and the authentication secret set on chaining, verifies the authentication_secret and the chaining_context for authenticity (by verifying its digital signature) and compares the enclosed integrity measurements with the current integrity measurements of the actual platform.
- Logging functions to log critical events to the SEP.
- Functions to access internal operation to initialize the PIM's internal integrity measurement chain by retrieving all integrity measurement values of components executed before the PIM has been executed.

### 2.4.3 Data Consistency and Plausibility

In order to verify the correctness of received data different checks can be done as classified in Figure 2.4. It is distinguished between station-based mechanisms thats require past information about a specific ITS station and message-based tests that are able to process unrelated V2X messages. Furthermore, this section focuses on the verification of consistency and plausibility of location-related information (position vector) of single-hop V2X neighbors. Most relevant information of the position vector are the absolute position associated with an absolute timestamp, heading, velocity, and acceleration. A message-based plausibility check of information is performed first to filter malformed data that violate predefined ranges of values. If the same piece of information is available multiple times in a message, a consistency check should be done subsequently. Related methods are detailed in Section 2.4.3.2. The received mobility data of neighbor stations should also be checked against locally available trusted first hand information as explained in Section 2.4.3.3 and 2.4.3.4. This data verification using local static knowledge can be done either on message basis or on station basis. Finally, received data can be compared with second hand information received from other stations. In case received information is not consistent with other received second hand information it might be challenging for the related mechanisms to interpret the results correctly, since both information sources are usually trusted equally. Mechanisms handling second hand information are discussed in Section 2.4.3.5.

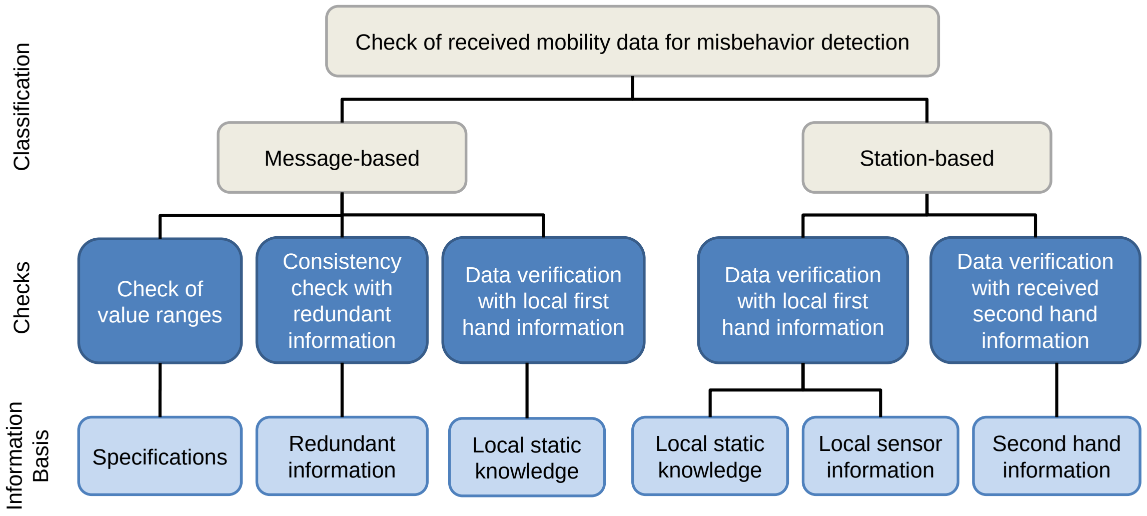

Classification
Checks
Information Basis
Check of received mobility data for misbehavior detection
Message-based
Station-based
Check of value ranges
Consistency check with redundant information
Data verification with local first hand information
Data verification with local first hand information
Data verification with received second hand information
Specifications
Redundant information
Local static knowledge
Local static knowledge
Local sensor information
Second hand information

Figure 2.4: Classification of data consistency and plausibility checks

In Figure 2.4 it is shown that the three checks on the left hand side are message-based and the two checks on the right hand side are station-based. The value range checks and the consistency checks are message centric and consider the messages separately. The data verification with received second hand information is in contrast station centric since previous messages have to be received that provide information about prior station behavior.

### 2.4.3.1 Message-Based Data Plausibility Checks

A message-based plausibility check is using predefined rules and physical boundaries. These checks are using a transmitted position vector that includes the position of the sender, its current speed and heading at a specific point in time. In these basic checks the given values of a position vector are compared with the predefined domain of definition.

The heading value shall follow the domain of definition according to related standardization for CAM and DENM as well as for network layer headers. A heading value larger than 360° for example should be considered to be not plausible. Furthermore, the velocity values shall be checked as well as the position of the sender. The position is usually encoded in the WGS84[2] format that includes a latitude and longitude value [8, 9]. For example, a velocity of a vehicle below -30 $\frac{m}{s}$ and beyond 130 $\frac{m}{s}$ is suspicious in normal road traffic.

#### 2.4.3.2 Message-Based Data Consistency Checks with Redundant Information

A message-based consistency check is possible if information is redundant, e.g. due to reception of multiple messages over different communication channels or due to redundant information on different layers of the communication stack. The general packet format of a V2X message as depicted in Figure 2.29 on page 71 shows that position information is available in different parts of a packet. Even though this position information is not equal due to possibly different interpretations on different layers, a comparison by means of consistency checks allows at least a detection of unexpected deviations. Large deviations consequently may indicate a misbehavior of the sender station if a malware has modified the position data on one layer only. However, it is necessary to be aware about variations between comparable information. For example, the position vector applied on one layer may be more inaccurate as the vector applied on another layer because in one case the raw GNSS signal is used and in the other case a dead reckoning optimized position is used. Another reason for variations could be a slightly different position reference point.

In order to additionally detect Sybil attacks the consistency of identifiers contained in V2X packets have to be checked. Therefore it is required that at least the node ID of the network header and the station ID of the payload are linked to the certificate coming as part of the security header according to the basic system standards profile of the Car-to-Car Communication Consortium (C2C-CC) [39]. In order to establish the linking the VSS creates a hash value from the currently used pseudonym certificate and uses parts of the value as certificate ID (cf. ETSI TS 103 097 [14]). This certificate ID is further used by the layers of the communication stack to derive their header specific identifiers. On packet reception the identifiers from the MAC header, network header, security header and payload are collected and finally compared on the top most message processing layer. If the IDs are not consistent or cannot be linked to the certificate or its certificate ID, the packet can be considered as malformed.

#### 2.4.3.3 Message-Based Data Verification with Local First Hand Information

By using static local first hand knowledge two different checks of the message content are considered that focus on the detection of replayed data.

- **Check of maximum communication range:** In a communication range check, the distance between the position of a single-hop sender and receiver is calculated. If this distance exceeds the maximum transmission range the location of the sender can be assumed to be not plausible. It is assumed that radios are used that follow the maximum specified transmission power according to IEEE 802.11p [23] and

[2]World Geodetic System 1984

ETSI ES 202 663 [6]. The mechanism was first mentioned by Golle et al. [18] and corresponds to the *Acceptance Range Threshold* sensor described by Leinmüller et al. [26]. In general, this kind of check aims to detect location-based replay attacks that are also known as tunnel or wormhole attack [21]. In this attack an attacker records an authenticated message at a location $l_1$, transmits it quickly to a location $l_2$ and re-broadcasts it at $l_2$.

- **Check of maximum transmission delay:** In addition to a distance check, the maximum transmission delay of single-hop messages can be verified by receiving ITS stations. According to ETSI TS 102 637-2 [8] the maximum transmission delay of CAMs shall not be larger than 100 ms. As a result, messages with an outdated timestamp or a future timestamp can be seen as not plausible. This kind of check is already part of emerging standards, i.e. IEEE 1609.2 [22] and ETSI TS 102 731 [7]. This check aims to detect time-based replay attacks where an attacker records a valid message at time $t_1$ and replays it later at the same location at a time $t_2$.

#### 2.4.3.4 Station-Based Data Verification with Local First Hand Information

In addition to the message-based checks, a station-based verification is reasonable using local first hand information. Two types of local first hand information are distinguished in the following. Static local knowledge about the network and its communication systems may be used to detect implausible behavior of adjacent stations. Furthermore, local sensors may be used to verify the location data of received messages.

**Checks Based on Static Local Knowledge** In this paragraph, four options are denoted that are based on static knowledge and standardized rules to check location-related data. These checks were first mentioned by Leinmüller et al. [26] and Schmidt et al. [32].

- **Check of maximum beacon frequency:** Since the wireless V2X channels are used cooperatively, the maximum transmission frequency of CAMs is limited. A plausibility check on the receiving station is able to count the received messages from the single-hop neighbors and is consequently able to detect violations according to the ETSI standard TS 102 637 [8, 9].
- **Check of suddenly appearing station:** In normal traffic conditions it can be assumed that new vehicles first appear at the boundary of the communication range. As a result, a first CAM from an ITS station with an unknown ID shall contain position data that states a certain distance between the sender's station and the receiver station. However, ID changes and hidden stations that might be caused by large buildings in urban environments require a context depended check of suddenly appearing stations.
- **Check of plausible movement:** Based on a physical mobility model for vehicles a position can be predicted using previously received position statements. When a new message is received, the predicted position can be compared with the stated

position whereupon large deviations are suspicious, hence may result in misbehavior detection. Since CAMs are broadcasted with a maximum frequency of 10 Hz [8], an accurate position vector of the next CAM can be assumed. By checking the movement plausibility, position jumps and unexpected mobility behavior can be detected.

- **Check of map related position:** A digital road map can be used to check the position of a sending vehicle station as long as the receiving ITS station is equipped with such a digital map. A digital road map may be required by traffic safety and efficiency applications anyway. However, a vehicle that cannot be assigned to a valid road segment of the local map is possibly driving on a private road or is parked beside a road. It has to be further considered that the local map may be outdated. Therefore the check might be used rather for additional validation of already suspicious behavior than as an independent check.

**Checks Based on Local Sensors** Stations that are equipped with local environment sensors can use their measurements to confirm or refute a stated location of a neighboring station. For example a local front radar transceiver is able to track different vehicles that are driving ahead of the own station. In the same way, other local distance and proximity sensors like cameras, lidar or infrared-based detectors can be used to check the stated position data of neighbors. Since front radar systems are already widely used in vehicles for autonomous cruise control, the plausibility checks may focus in the first stage on applying a radar transceiver as local sensor. The concept of using local sensors to verify stated locations in V2X communications has been first comprehensively discussed by Yan et al. [43] and was subsequently used within other related concepts [17, 32].

- **Radar approved position:** If a received position of a neighbor station can be mapped to a radar object of the local sensor, then this vehicle position information can be assumed to be trustworthy. However, RSUs are in general not confirmable with a radar sensor.
- **Radar conform position:** In addition, the object detection of a local radar can be used to refute a stated location. If a neighbor vehicle claims a position that is located between the own station and an object that is detected by the radar, then this vehicle position is not trustworthy. Assuming vehicles trust their own sensors and on-board networks, a detected false position claim can be trusted. If however received second hand information is used to check the plausibility of received position claims the verification might not be trustworthy as discussed in Section 2.4.3.5.

#### 2.4.3.5 Node-Based Data Verification with Received Second Hand Information

A station that receives conflicting – but equally trusted – information from two different stations cannot directly determine which statement is true and which is false. However, by collecting additional information about the same or a similar statement from different independent senders, the receiver may be able to take a decision assuming that the majority of provided information is correct.

- **Check of vehicle overlaps:** Since vehicles are periodically broadcasting CAMs with their absolute position and their rough stations' dimensions, a check of position overlaps can be performed by comparing the locations of near-by stations. In [1] the vehicle overlap check is further detailed.
- **Neighborhood table exchange:** As discussed in the related work [26] and [42] neighbors may distribute their local first hand information (e.g. radar tracked stations) or reputation information about their neighbor stations. A receiver of this information is able to compare the received tables with other received tables and with its local neighbor information. However, all received data from neighbors can be equally trusted as long as valid cryptographic credentials are used to sign the messages. Therefore, attackers would also be able to distribute faked neighbor tables. Moreover, the exchange of additional data for security purposes would increase the load on the wireless channel dramatically. According to Schoch [33] the reactive exchange of position information creates unacceptable communication overhead and the verification does not profit much or even suffers from it. Increasing the load of the wireless V2X communication channel is critical since the security overhead is already substantial due to relatively large security credentials [2, 14, 22]. Additionally, the exchange of security-related data may create new vulnerabilities and attack vectors that could be misused. As a result, we argue to avoid or at least minimize the amount of additional redundant data that are transmitted for plausibility checks.

In the end, the various consistency and plausibility need to be integrated into a common framework for misbehavior detection where detectors can be added in a flexible way. Since this is a topic of on-going research, we present some preliminary results in our WP5 work.

### 2.4.4 Privacy Protection

As PRESERVE aims at securing the V2X communications (particularly with regard to CAMs and DENMs) privacy should be considered as well. However, CAMs broadcast information (location, speed, heading, station ID, etc.) that may reveal the real driver identity, or enable tracking attacks. Moreover, as CAMs and DENMs are signed with certificates that may identify the driver or the vehicle. Thus, there is a conflict between privacy and authentication.

To ensure location privacy of drivers despite broadcasting CAMs and DENMs, PRESERVE uses pseudonyms. Pseudonymity is a mechanism to hide the real identity of the sender. However, using pseudonyms is only efficient if their lifetime is limited. Therefor, the pseudonyms must be changed during the vehicles' lifetime. Unfortunately, changing pseudonyms periodically or randomly is not efficient, and does not provide sufficient location privacy according to [41]. As a result, PRESERVE investigates pseudonym change strategies to carefully select the best-suited one as part of our WP5 research work.

In PRESERVE, the issue of pseudonym management is addressed in Section 2.4.4.1, which includes pseudonym provisioning to vehicles and pseudonym change. We recommend to use a *pseudonym change block* mechanism in order to prevent a pseudonym change during critical situations. For example, a rugged vehicle should not change its pseudonym after the first warning because following vehicles would consider the two warnings as different, and thus, believe that there are two vehicles in danger. Moreover, the use and frequent change of pseudonymous IDs could lead into unreachable vehicles during the pseudonym change phase. That is why PRESERVE will analyze the impact of pseudonymity on the connectivity during the hybrid FOT.

Another important aspect is that the pseudonym should hide all vehicle identifiers, i.e. MAC address, network layer node identifier, and station ID contained in CAMs and DENMs. But these changes might provoke inconsistencies in routing tables for example. This issue is considered by the PRESERVE VSA and an approach is discussed in Section 2.4.4.3. Besides handling of pseudonym changes, the VSS API provides commands for pseudonym change blocks which can be used by applications in critical traffic situations as describe in Section 2.4.4.4.

To enforce privacy policies for data access, we proposes to follow the PeRA architecture defined in the PRECIOSA project [24]. The PeRA subsystem as further detailed in Section 2.4.4.2 permits to ensure that data processing of personal data is done according to the current privacy policy of the vehicle's driver. The PeRA component can also be used to authorize broadcast communication for specific third party applications.

To ensure privacy in the backend such as the PKI, the VSS receives a long-term certificate and short-term pseudonyms from different CAs as described in more detail in Chapter 3.

#### 2.4.4.1 Pseudonym Management Module

The Pseudonym Management Module (PMM) is responsible for the administration of the ITS station's pseudonym certificate pool. Pseudonyms are a set of distinct certified public keys that do not provide identifying information. As introduced in Section 1.2 each ITS station is equipped with a set of short-term identifiers (namely pseudonyms) that consist of a key pair and corresponding certificates that are issued by a CA. Those pseudonyms are similar to the long-term identifiers (stored at the ID & Trust Management Module described in Section 2.5.1.1) with the exception that they do not include any information identifying a device, vehicle, or individual. The tasks of this component can be summarized as:

- requesting the CA for new pseudonym certificates when the pool of valid PCs is or is almost empty,
- deleting pseudonyms from the pool (e.g. when they are no longer valid),
- monitoring the pseudonym usage status.

The PMM decides how long a pseudonym is allowed to be used in V2X communications and when to change to another pseudonym, according to the privacy policy. At initialization time, the PMM is configured by the security manager, in particular for the pseudonym refill policy.

The private keys of the pseudonyms are generated and stored inside the HSM by using the cryptographic services. PMM only manages the public keys and related certificates of the pseudonyms as well as their handle to the private key. Since the PMM is on the critical path of outgoing messages, the latency has to be as low as possible for requesting the active pseudonym as defined in the requirements analysis of PRESERVE [36, Section 3.1.8].

The PPM is also responsible for the coordination of changing identifiers in the layers of the communication stack by triggering a pseudonym change as described in the process in Section 2.4.4.3. The trigger for pseudonym changes is probably not frequently used (approximately every few minutes). Nevertheless, the process of changing the pseudonym is very critical with respect to timing and delay and should be done in less than 20 ms.

Requests for new pseudonym certificates sent to the CA are created by the ID & Trust Management Module (IDM) and are signed with the long-term identifier of the ITS station. The related process is described in Section 2.5.1.3.

As illustrated in Figure 2.5 the PMM is connected inside the VSS with the Security Manager (SM), Secure Communication Module (SCM), the ID & Trust Management Module (IDM), and the Cryptographic Services (CRS).

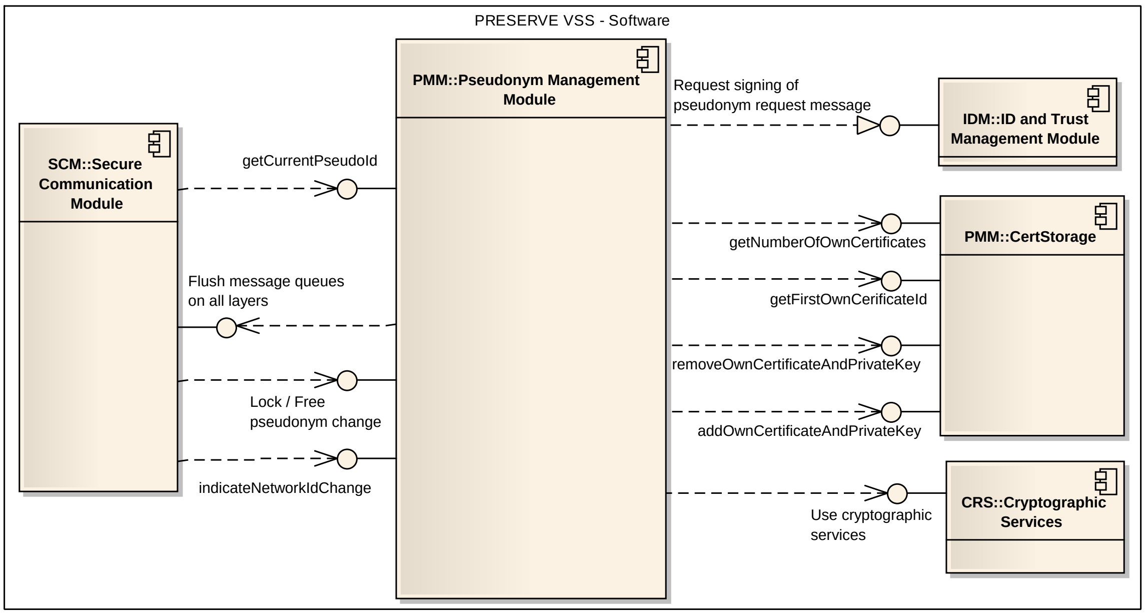

PRESERVE VSS - Software
PMM::Pseudonym Management Module
SCM::Secure Communication Module
getCurrentPseudoId
Flush message queues on all layers
Lock / Free pseudonym change
indicateNetworkIdChange
Request signing of pseudonym request message
IDM::ID and Trust Management Module
PMM::CertStorage
getNumberOfOwnCertificates
getFirstOwnCerificateId
removeOwnCertificateAndPrivateKey
addOwnCertificateAndPrivateKey
Use cryptographic services
CRS::Cryptographic Services

Figure 2.5: Integration of the PMM into the on-board security architecture

In the following, examples of internal data flow between the PMM and other internal VSS components are given as well as the dependencies to related components.

**Data Flow**

- The convergence layer uses the interface of the PMM in order to block the pseudonym change for a defined time and to release this block.
- When the SCM gets outgoing V2X messages that have to be signed with the correct and active pseudonym, it asks the PMM to provide the appropriate pseudonym.
- The PMM manages the correct pseudonym change on the communication layer as described by the sequence diagram shown in Figure 2.8 on page 38.
- The PMM is responsible to trigger the request for new pseudonyms when it detects that the number of pseudonyms is under a specified threshold (in its configuration). For this process, it calls the ID & Trust Management module.
- The PMM requires the GNSS synchronized time from the VSS platform in order to check the expiry of own certificates and certificates sent by neighboring ITS stations.

**Dependencies** The PMM strongly depends on:

- ID & Trust Management in order to create and send new pseudonym requests
- The configuration module for its privacy policy and all parameters that control its behavior

#### 2.4.4.2 Privacy-enforcing Runtime Architecture

The privacy subsystem manages applications that require detailed access to personal information, such as credit card details of a user. It is based on the results of the PRECIOSA[3] project. The main building block of the privacy subsystem is the Privacy-enforcing Runtime Architecture (PeRA). In the following it is explained how PeRA can be employed to enforce privacy requirements of users.

PeRA is an architecture that is able to enforce privacy protection in the complete system at runtime. The Query API and the Importer/Exporter are responsible for interfacing with outside components (e.g. sensors, communication stack) and applications. The privacy control monitor, as described in the PRECIOSA deliverable D7 [24, Section 5], is the main component.

PeRA supports two kinds of applications. External applications can make queries to PeRA in order to access sensor information or other personal information managed in PeRA's secure database. Applications that require privacy-policy-compliant communication to other external entities (e.g., "Point of Interest notification with additional information exchange e.g. booking of hotel on the road" as described in the PRESERVE deliverable D1.1 [36, Section 1.2.2.4]) should be implemented as controlled applications and run completely inside the PeRA subsystem. Applications that do not require fine-grained policy control and have undergone a Privacy Impact Assessment (PIA) can work without PeRA interaction.

[3]http://www.preciosa-project.org/

**Connection to other Modules of the PRESERVE Architecture** PeRA interacts with the Platform Integrity Module (PIM), the Convergence Layer (CL), and the Communication Control Module (CCM) in order to realize incoming and outgoing information flow for PeRA-controlled applications as shown in Figure 2.6. PIM is used to ensure that all PeRA components are in a certified state and have not been tampered with. For applications

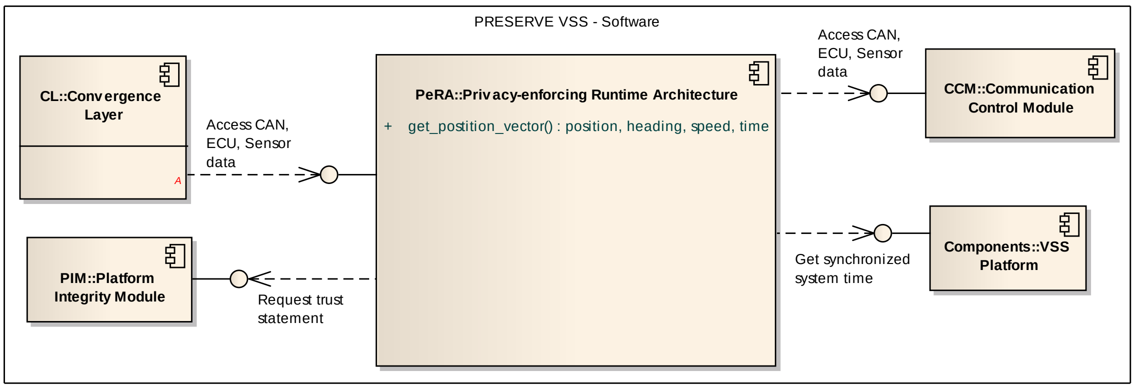

PRESERVE VSS - Software
CL::Convergence Layer
Access CAN, ECU, Sensor data
PeRA::Privacy-enforcing Runtime Architecture
+ get_postition_vector() : position, heading, speed, time
Access CAN, ECU, Sensor data
CCM::Communication Control Module
PIM::Platform Integrity Module
Request trust statement
Get synchronized system time
Components::VSS Platform

Figure 2.6: Integration of PeRA into the on-board security architecture

that require access to local sensor data, PeRA's Importer/Exporter module requests the necessary data from the CCM. For outgoing communication, the CL is used to send data packets to external entities. Likewise, applications outside of PeRA's control, which need to access personal information stored inside the PeRA-controlled secure database, can pose queries to the Query API via the Convergence Layer.

**Data Flow** Figure 2.7 shows the interaction of external applications with PeRA in order to access for example location data. Instead of accessing on-board sensors or the CAN

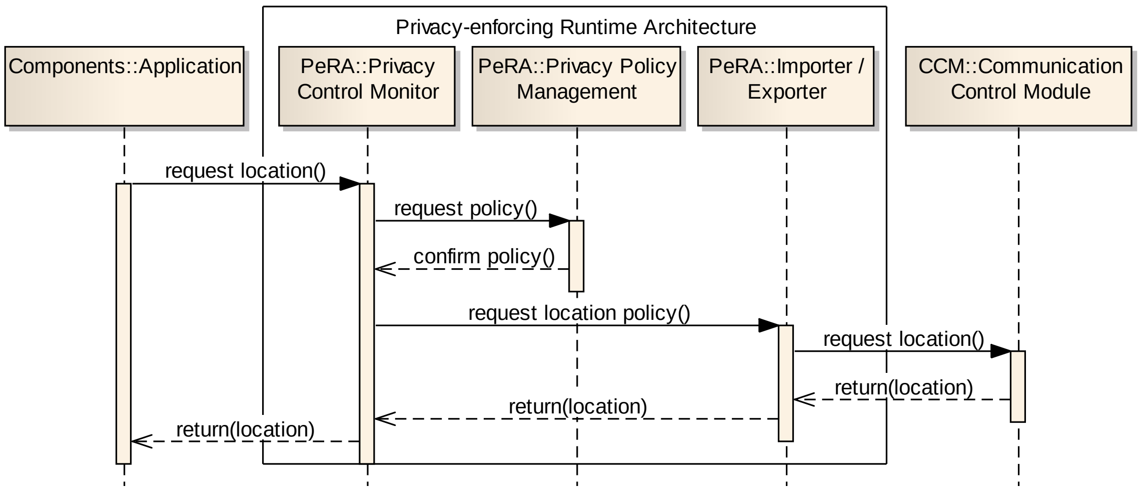

Privacy-enforcing Runtime Architecture
Components::Application
PeRA::Privacy Control Monitor
PeRA::Privacy Policy Management
PeRA::Importer / Exporter
CCM::Communication Control Module
request location()
request policy()
confirm policy()
request location policy()
request location()
return(location)
return(location)
return(location)

Figure 2.7: Access of privacy related data via Privacy-enforcing Runtime Architecture

bus directly, these applications use PeRA for data access. Especially, access to privacy-relevant information such as the current location is controlled by PeRA, and privacy policy compliance is mandated by the Privacy Control Monitor. An application requests, for example, the current location via Convergence Layer and the Query API from PeRA. The Privacy Control Monitor checks the request for policy compliance, interacting with the Privacy Policy Manager to obtain policy couplings, and forwards it to the Importer/Exporter, which connects the CCM in order to access the CAN bus or directly the GNSS receiver. If other stored data is required, it is taken from a PeRA secure data storage or meta data repository.

Query results are returned to the application via the same path. Possibly, data manipulation (e.g., anonymization or fuzzyfication) are applied to comply with privacy policies. Likewise, an application using PeRA never communicates with the network directly, but channels send requests via the Query API. For the processing of the privacy relevant operations, the access to the Platform Integrity Module is used to check the integrity of the PRESERVE VSS. In case of a compromise the PeRA may reject privacy related requests.

**Timing** PeRA is primarily used by applications which exchange privacy-policy-annotated data with external entities. Only if general V2X message content is requested via PeRA (e.g. location information for CAMs or DENMs) strong latency requirements are given as defined in the requirements analysis of PRESERVE [36, Section 3.1.8]. Otherwise, no strict performance and timing requirements are defined.

#### 2.4.4.3 Change of Pseudonymous Identifiers

Changing the ID of an ITS station entails a relatively complex process. The main challenge is to change the ID on all involved communication layers at the same time. In other words, a data packet transmitted over the air must not contain identifiers of different related to pseudonyms. Nevertheless, it is not necessary from privacy protection point of view that all IDs used on the different communication layers can be assigned to a specific pseudonym certificate. In order to check the packet data consistency on receiver side it is, however, required to link on sender side relevant IDs to the pseudonym certificate, cf. Section 2.4.3.2. The main challenge discussed in this section is to avoid that an ID used with an old pseudonym certificate is available in a packet that is secured by using a new pseudonym certificate. For example, it must not happen that the layer X adds a new ID to its packet header or payload as long as layer Y is still adding the old ID.

The process for changing a pseudonym shown in the sequence diagram in Figure 2.8 is considering this requirement. It must be considered that only outgoing packets are involved in the following process. Incoming packets are not affected.

At first, all the different layers that are adding IDs to CAMs or DENMs have to register themselves at the VSS. For this task, the API at the convergence layer provides an interface which is also able to identify the registered layer by using a local configuration.

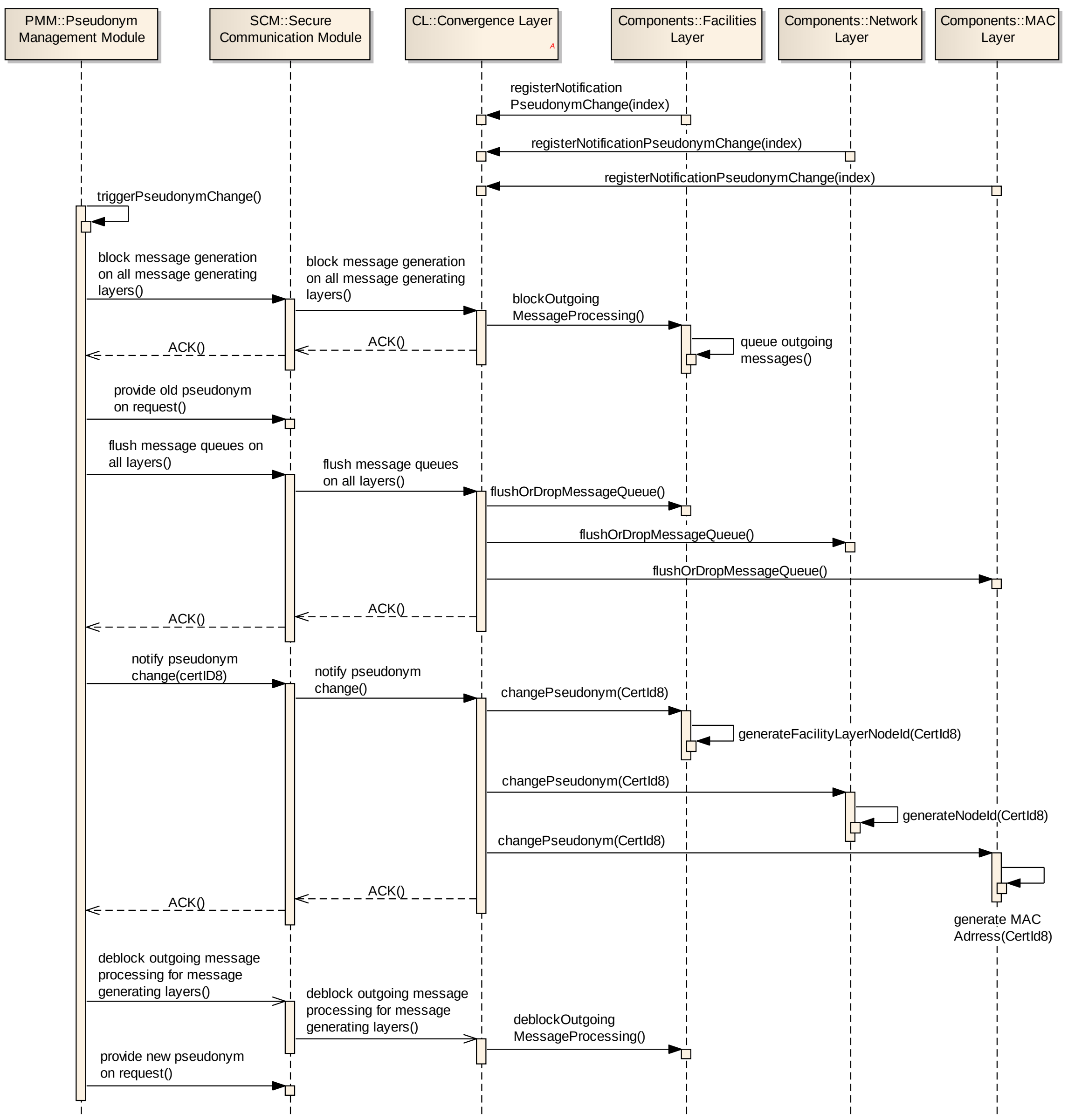

PMM::Pseudonym Management Module
SCM::Secure Communication Module
CL::Convergence Layer
Components::Facilities Layer
Components::Network Layer
Components::MAC Layer
registerNotification PseudonymChange(index)
registerNotificationPseudonymChange(index)
registerNotificationPseudonymChange(index)
triggerPseudonymChange()
block message generation on all message generating layers()
block message generation on all message generating layers()
blockOutgoing MessageProcessing()
queue outgoing messages()
ACK()
ACK()
provide old pseudonym on request()
flush message queues on all layers()
flush message queues on all layers()
flushOrDropMessageQueue()
flushOrDropMessageQueue()
flushOrDropMessageQueue()
ACK()
ACK()
notify pseudonym change(certID8)
notify pseudonym change()
changePseudonym(CertId8)
generateFacilityLayerNodeId(CertId8)
changePseudonym(CertId8)
generateNodeId(CertId8)
changePseudonym(CertId8)
generate MAC Adress(CertId8)
ACK()
ACK()
deblock outgoing message processing for message generating layers()
deblock outgoing message processing for message generating layers()
deblockOutgoing MessageProcessing()
provide new pseudonym on request()

Figure 2.8: Change of Pseudonyms

Inside the Pseudonym Management Module (PMM), it is necessary to know the order of the registered layers in order to synchronize the notification later on.

As soon as a pseudonym change is triggered by the PMM, all registered message generating layers have to be informed about stopping their outgoing message processing. Affected layers may queue newly generated messages or drop new messages but it is important that new messages are not transmitted downwards to lower layers of the communication stack. When the PMM has received an acknowledgment from all message generating layers, all registered layers are informed about flushing or dropping their message queues one after another. At first the highest layer (i.e. facilities layer) is notified and subsequently the network layer and access layer, respectively.

As long as the pseudonym change is triggered but not finished, the old pseudonym must be used by the VSS in order to complete open security processes that are related to the old pseudonym's private and public key. That means, that the old pseudonym private key is used to sign outgoing messages that are stored in the queues of the layers.

When all messages with the old identifier are processed and sent then the VSS notifies all registered communication layers at the same time to change their ID. The notification contains the CertId8 related to the new pseudonym certificate. According to IEEE 1609.2 [22, Section 5.2.6] and ETSI TS 103 097 [14] the CertId8 shall be calculated by hashing the encoded certificate with SHA-256 and taking the low-order 8 bytes of the hash output. If it should be possible to link monitored messages to a specific ITS station, this CertId8 shall be used by the different layers in order to calculate their layer specific ID.

After receiving an acknowledgment from all involved layers, the message generating layer is informed to unlock outgoing message processing as shown on the bottom of Figure 2.8. At the same time the old certificate with its private and public key should be marked as used inside the HSM and substituted by the new certificate key pair. Additionally, the PMM has to provide this new certificate to the requesting modules (e.g. Secure Communication Module) if outgoing messages should be extended by a security header that contains a pseudonym certificate.

**Reusing Pseudonym Certificates** The old pseudonym certificate with its related private and public key may be reused later if no unused pseudonyms are available inside the PMM and the certificate is still valid. Nevertheless, the strategy for reusing pseudonym certificates may be configurable and change according to the interval of requesting new pseudonym certificates from the PKI and the privacy policy used.

**Processing of Stored DENM Messages** It may be happen that V2X messages (i.e. DENMs) are valid for a longer time period. If an application creates a notification that should be presented frequently to neighboring ITS stations for a defined period of time, the message is created only once and will be send afterwards several time. Therefore, the ID inside the V2X message payload is fixed to the time of generation. As discussed in this section it is necessary to have consistent IDs on the different layers. Due to the privacy protection requirement it is not allowed to make a pseudonym change as long as it can be expected that such kind of messages are sent out.

In order to consider this issue a simple solution should be followed in the VSA. The application that is responsible for managing mentioned DENMs has to lock the pseudonym change at the VSS as long as it can be expected that the V2X message will be sent. For example, the application can request a pseudonym change lock for the validity time of the DENM according to the process described in Section 2.4.4.4.

#### 2.4.4.4 Lock Change of Pseudonymous Identifiers

On the one hand it is necessary that pseudonym changes are not triggered by the PMM if messages with old IDs are sent continuously as discussed in Section 2.4.4.3. On the other hand applications may not work correctly if adjacent ITS stations change their IDs in specific, possibly dangerous, situations. In the second case, a V2X application requires that all stations that are involved in a critical situation stick to their ID. As such a situation is mostly related to a specific geographic location, all involved ITS stations with their V2X applications may be able to identify the critical situation independently.

As soon as a V2X application has identified that the own station is inside a critical area or a DENM is available that may be sent out in the future with unchanged IDs, the pseudonym change has to be blocked at the VSS. The process of blocking / locking the pseudonym change via the API of the PRESERVE VSS is presented in the sequence diagram shown in Figure 2.9 and the activity diagram shown in Figure 2.10.

The pseudonym change locks are managed by the PMM. Every application or entity of the communication layer is able to use the respective method of the convergence layer in order to request a lock and release it afterwards. Every requester is identified by an 2 Byte *requesterID* that should be unique inside the ITS station architecture. Ensuring the uniqueness is not part of the VSS. As shown in Figure 2.9 the requester (e.g. a V2X application by its own or the facility layer as proxy) requests a pseudonym change lock by providing its unique identifier and the desired duration for this lock in milliseconds.

As illustrated in the activity diagram shown in Figure 2.10, the PMM then checks first whether the requested duration is smaller than possibly available locks from other requesters. If this is the case, the requester gets an acknowledgment with its desired duration in the confirmation of the pseudonym lock request. Additionally, the requesterID with its duration time is stored inside the PMM. If the requested duration is larger than all other stored pseudonym change locks, the PMM checks if the desired end of the lock lays inside the validity time of the active pseudonym certificate. Having a pseudonym certificate that expires before the requested duration ends, the requester gets in its confirmation the duration in milliseconds until certificate expiry (*certificate expiry time - current time - buffer time*). In this case inside the PMM a timer is started that triggers the next pseudonym change. In the other case, if the desired duration is smaller than the certificate expiry time, another timer is started with the requested duration that unlocks the pseudonym change as long as there is not a new request received before the timer expiration time is exceeded.

If a V2X application detects that the own ITS station is outside of a critical situation but the requested pseudonym change lock has not yet expired, the application shall notify the VSS. In this case, the application or the facilities layer uses the method *unlockPseudonym-Change* at the API of the convergence layer with the same requesterID used previously in the lock. After receiving a pseudonym change unlock, the PMM checks whether the requester is stored in the internal list and deletes it. Then the PMM checks if other stored requests had requested a longer duration than the just unlocked one. If this is the case,

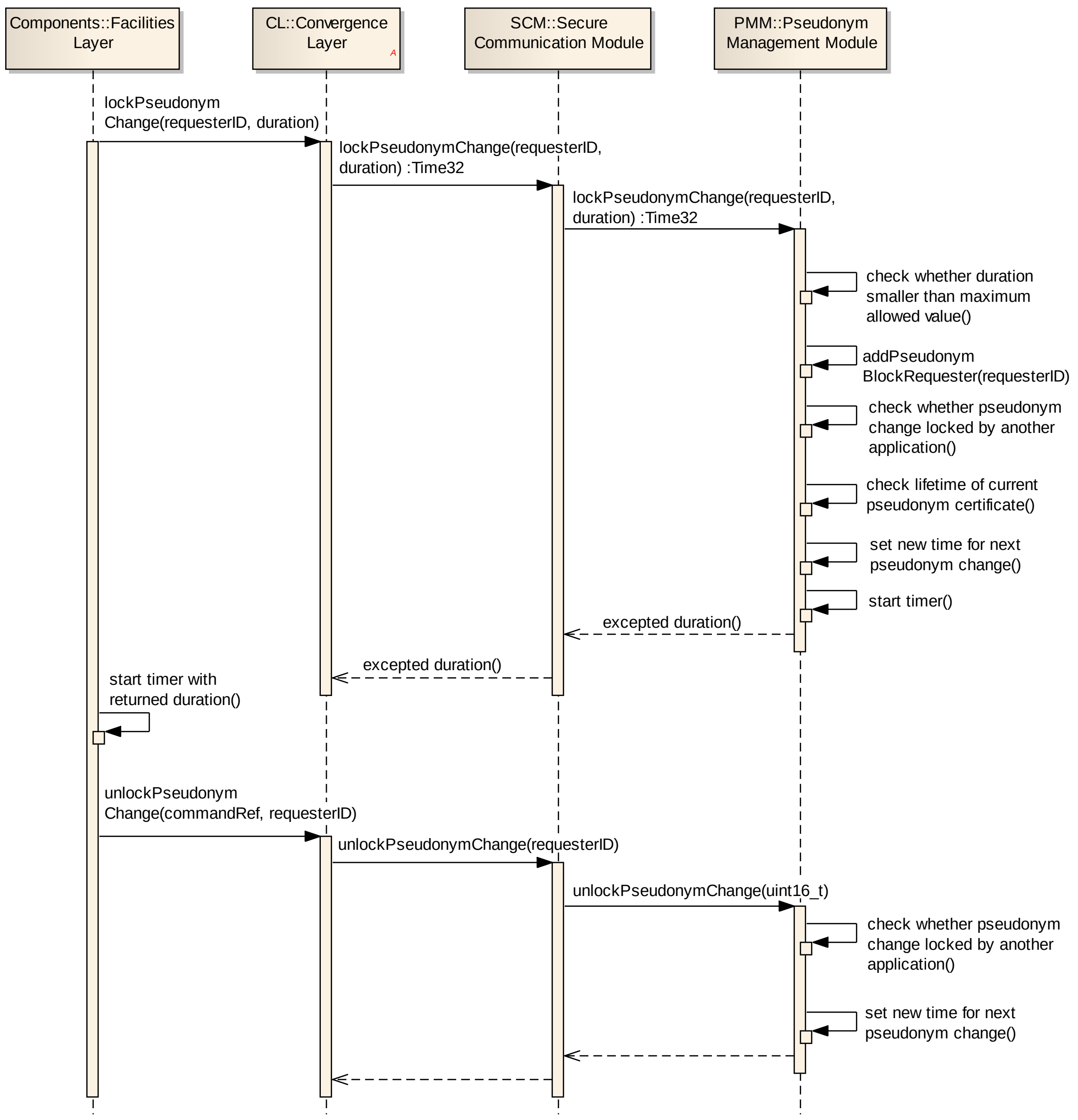

Components::Facilities Layer
CL::Convergence Layer
SCM::Secure Communication Module
PMM::Pseudonym Management Module
lockPseudonym Change(requesterID, duration)
lockPseudonymChange(requesterID, duration) :Time32
lockPseudonymChange(requesterID, duration) :Time32
check whether duration smaller than maximum allowed value()
addPseudonym BlockRequester(requesterID)
check whether pseudonym change locked by another application()
check lifetime of current pseudonym certificate()
set new time for next pseudonym change()
start timer()
excepted duration()
excepted duration()
start timer with returned duration()
unlockPseudonym Change(commandRef, requesterID)
unlockPseudonymChange(requesterID)
unlockPseudonymChange(uint16_t)
check whether pseudonym change locked by another application()
set new time for next pseudonym change()

Figure 2.9: Lock change of pseudonyms

the PMM checks against collisions between pseudonym certificate validity and available lock and sets afterwards the timer respectively.

## 2.5 Security Management

The Security Manager is responsible for the configuration and management of all the components that compose the security stack. At run time it instantiates all the security modules, for example CRS, PMM, IDM, SCM. For each security module which needs to

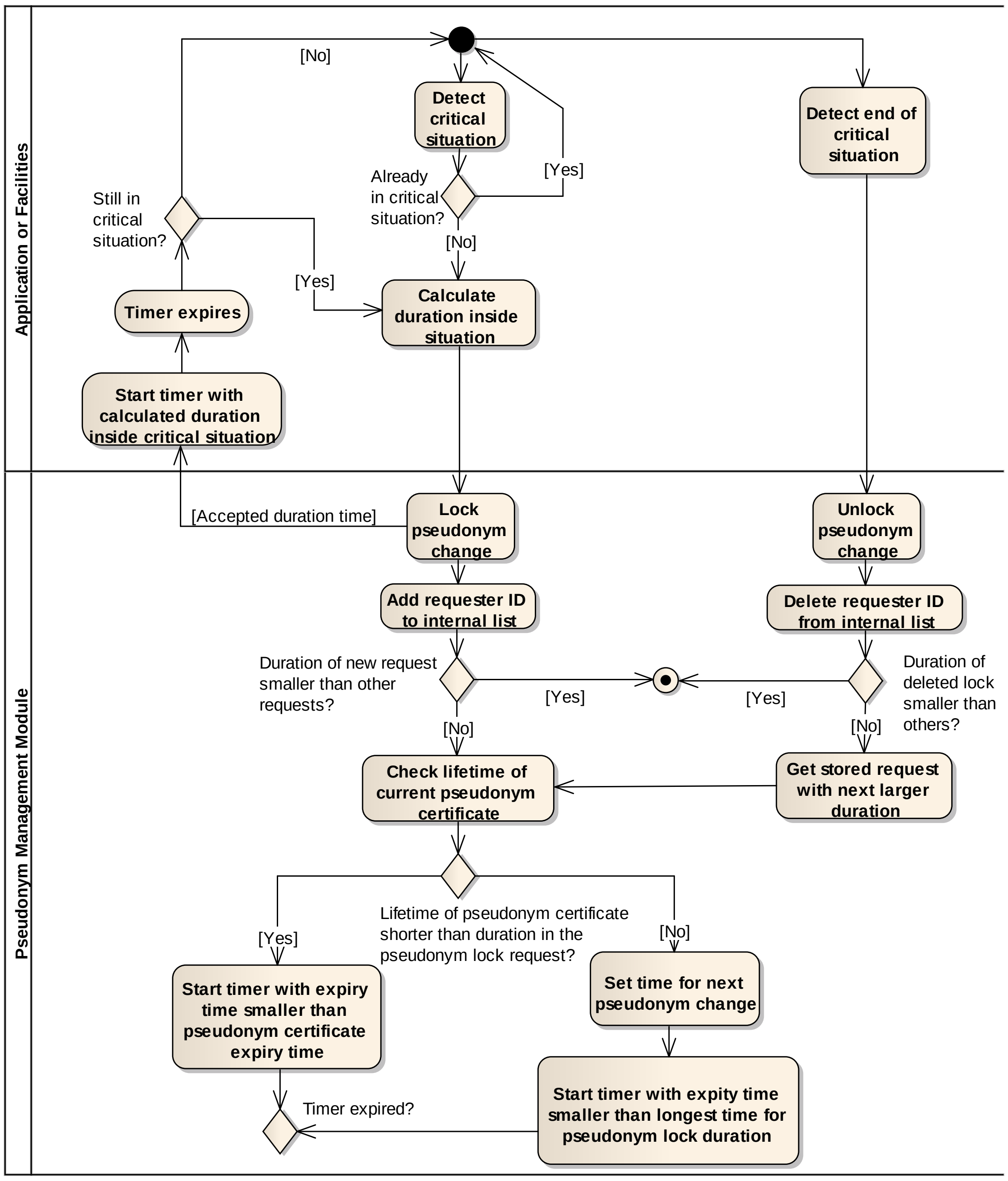

Application or Facilities
Pseudonym Management Module
[No]
Detect critical situation
Already in critical situation?
[Yes]
[No]
Detect end of critical situation
Still in critical situation?
[Yes]
Timer expires
Calculate duration inside situation
Start timer with calculated duration inside critical situation
[Accepted duration time]
Lock pseudonym change
Unlock pseudonym change
Add requester ID to internal list
Delete requester ID from internal list
Duration of new request smaller than other requests?
[Yes]
[Yes]
Duration of deleted lock smaller than others?
[No]
[No]
Check lifetime of current pseudonym certificate
Get stored request with next larger duration
Lifetime of pseudonym certificate shorter than duration in the pseudonym lock request?
[Yes]
[No]
Start timer with expiry time smaller than pseudonym certificate expiry time
Set time for next pseudonym change
Timer expired?
Start timer with expity time smaller than longest time for pseudonym lock duration

Figure 2.10: Activity diagram of locking the pseudonym change

interact with another security module it passes to the former a pointer to the latter. Then it reads from a file the configuration for these modules.

The Security Manager is connected with the following modules of the security stack:

- Cryptographic Services (CRS)

- Pseudonym Manager Module (PMM)
- ID & Trust Management Module (IDM)
- Secure Communication Module (SCM)

For the other security management tasks like credential management and security entities management separate modules are designated in the PRESERVE architecture that are detailed in the following Sections 2.5.1 and 2.5.2.

### 2.5.1 Credential Management

This section shows how identities and credentials are managed in the V2X on-board security architecture. In Section 2.5.1.1 the ID and trust management module is described that is responsible for this management. Section 2.5.1.2 describes subsequently how long-term certificates and pseudonyms are requested when they are no more valid or when there are not enough pseudonym certificates available in the ITS station.

#### 2.5.1.1 Identification and Trust Management Module

The Identification & Trust Management Module (IDM) defines the components that are responsible for handling identities and credentials, with various operations including provision, renewal, or revocation of credentials. It manages the CA certificates that are used for the verification of other certificates and pseudonym certificates. For its own pseudonyms, when their number falls under a specified threshold (this situation is detected by the PMM module), this module is responsible for requesting more pseudonyms from a CA. It also manages the IDs of neighboring nodes. For every ITS station in the neighborhood an entity is managed that contains information about e.g. certificate ID, issuer, validity of pseudonym certificate and station trustworthiness. When some information is missing for completing the verification of received messages, the IDM will request this missing information. The process is as follows. Incoming signed messages can contain either the complete certificate of the signer or only the digest of this certificate (for performance reason). In the latter case, it may happen that the corresponding certificate is not known by the receiving ITS station. When this situation happens, the next signed message sent by the ITS station contains a request for the missing certificate. In the first case, during the verification of the certificate chain, it may happen that the certificate of the CA is not known by the ITS station. Again, when this situation happens, the next signed message sent by the ITS station contains a request for the missing certificates (the certificate of the CA and the certificate of the sender). The CRS module detects these situations and informs the IDM module which is responsible for taking the appropriate decision during the next sending. This mechanism is also specified in the ETSI standard TS 103 097 [14].

Besides the management of neighboring nodes, the IDM is responsible for the management of its own long-term certificate and its related ID. If this long-term certificate appears to be invalid, the IDM requests a new long-term certificate from a CA.

The IDM as displayed in Figure 2.11 depends strongly on the CRS for all the cryptographic operations (e.g. verifying the signature of certificate responses). On the other hand, the PMM depends on the availability of the IDM. The Security Event Processor in contrast does not depend on the IDM but may use its interfaces if available.

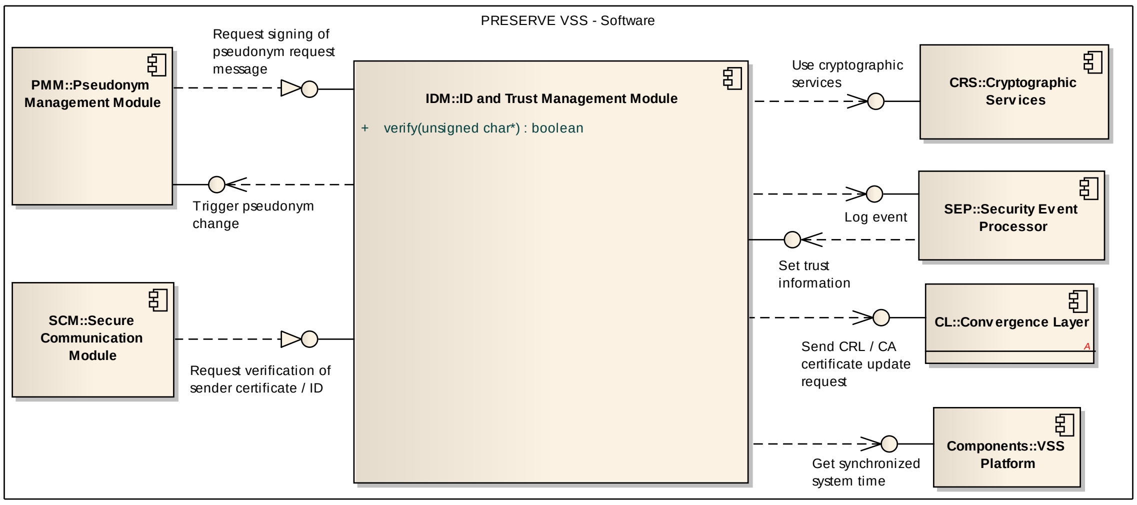

PRESERVE VSS - Software
PMM::Pseudonym Management Module
Request signing of pseudonym request message
Trigger pseudonym change
IDM::ID and Trust Management Module
+ verify(unsigned char*) : boolean
SCM::Secure Communication Module
Request verification of sender certificate / ID
Use cryptographic services
CRS::Cryptographic Services
Log event
SEP::Security Event Processor
Set trust information
CL::Convergence Layer
Send CRL / CA certificate update request
Get synchronized system time
Components::VSS Platform

Figure 2.11: Integration of the IDM into the on-board security architecture

In the following, some examples of internal data flow are given followed by related timing requirements.

### Data Flow

- V2X messages that are received by the ITS station must be verified. At first the SCM module is requested to check the validity of the sender's pseudonym certificate. If the sender's authentication is verified, then with the help of the Cryptographic Services (CRS) the signature of the message is verified. The IDM Module manages the trustworthiness of the sender certificate in order to avoid multiple cryptographic verifications of the same certificate.
- A signed V2X message that is received by the ITS station may contain only the digest of the pseudonym certificate used for signing the message. If the certificate that corresponds to this digest is not known the CRS module will inform the IDM module that the certificate is missing. Therefore a request for obtaining the missing certificate will be attached by the IDM module to the next message that is sent. A similar mechanism is used if the CA certificate is detected as missing by the CRS module during the verification of the certificate chain. In this case, several certificates will be requested by the IDM module: the pseudonym certificate of the sender and the certificate of the CA that was used to sign the pseudonym certificate of the sender.
- The IDM should detect collisions between its own IDs / addresses and IDs / addresses from received messages. In case of an approved collision an immediate

pseudonym change should be triggered. Nevertheless, a collision detection must consider misbehavior and therefore has to compare its own certificate with the certificate of the message sender before triggering a pseudonym change.

- As stated above, when the Pseudonym Management Module (PMM) detects that it has not enough pseudonyms left, it will request the IDM to download more pseudonyms from a PKI. The IDM uses the long-term certificate for signing the request. When it receives a successful response from the PKI, it stores the pseudonyms with the help of the CRS module.
- For all signing processes with the long-term ID, the IDM accesses the Cryptographic Services (CRS) in order to sign pseudonym requests or generate key pairs for the long-term ID.
- A synchronized time is needed for checking expiry of own certificates and pseudonym or CA certificates from neighboring ITS stations.

**Timing**

- The IDM Module is on the critical path of incoming messages as shown in the sequence diagram in Figure 2.21 on page 60. Therefore, low latency requirements are given as defined in the analysis of PRESERVE deliverable D1.1 [36, Section 3.1.8].
- Signing pseudonym refill requests has maybe also specific timing and minimum latency requirements. In case of transmitting the pseudonym refill request via ITS-G5A communication link, the full process must be finished in less than approximately 20 seconds. It is estimated that a vehicle is approximately 20 seconds within the communication range of a RSU in urban environments.

#### 2.5.1.2 Long-term Certificate Request

As described in Section 1.2, the V2X message protection is realized by using digital certificates in order to sign outgoing packets and verify them at the receiving ITS station. Two types of certificates are used in the PRESERVE VSA: long-term certificate and pseudonym certificate. The sequence diagram in Figure 2.12 provides an overview of getting a long-term certificate from the PKI.

The presented process must be executed only once at the initialization of the VSS or the ITS station. In general, the process can further be split into two stages.

1. In the first stage, the VSS generates an ECDSA key pair that is used later as *Device Identity Key* (IDK). A unique module-id and the public part of the IDK is transmitted over a trustworthy channel to the LTCA. As this process reflects the initial registration (bootstrapping) of the VSS, it is important that the transmission of this information cannot be done by arbitrary requester. It must be omitted that an attacker registers its own system as valid VSS. Further details regarding this first stage can be found in the PKI Memo document of the C2C-CC [34, Section 6.1].

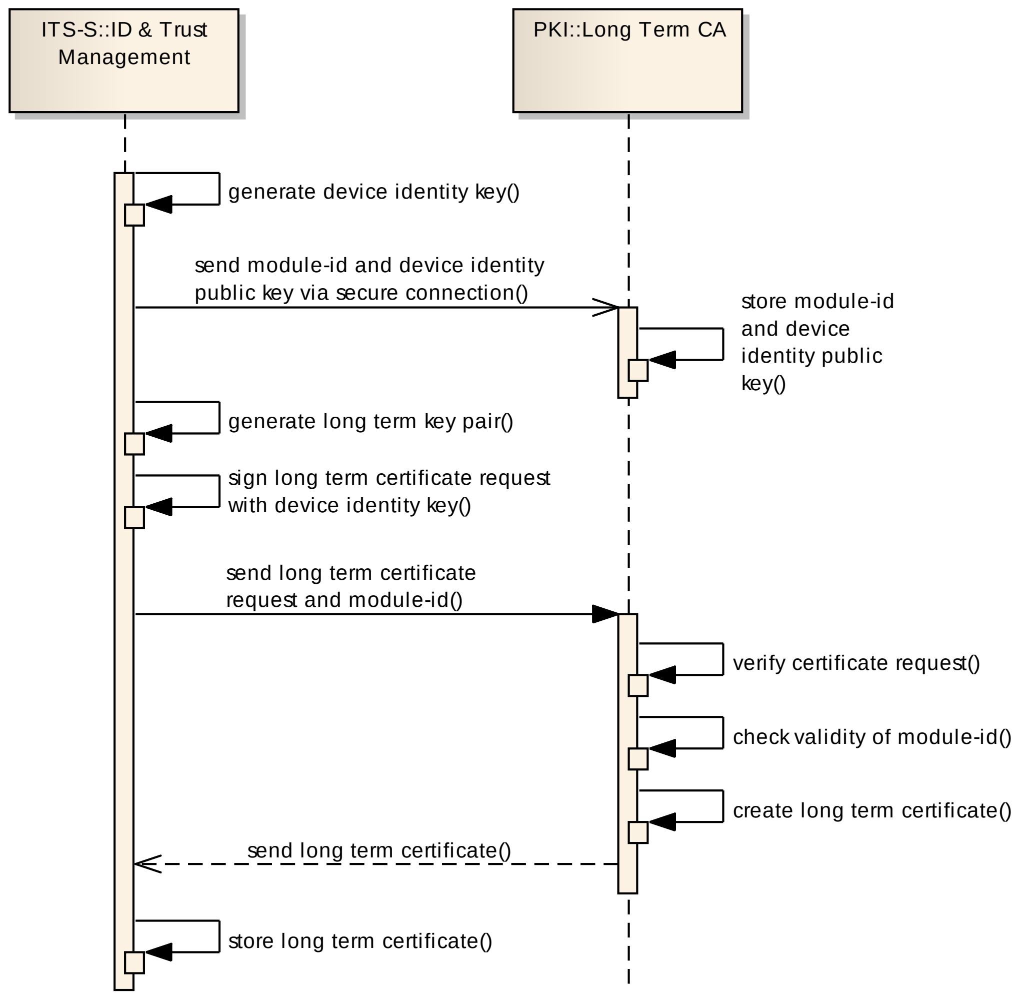

ITS-S::ID & Trust Management
PKI::Long Term CA
generate device identity key()
send module-id and device identity public key via secure connection()
store module-id and device identity public key()
generate long term key pair()
sign long term certificate request with device identity key()
send long term certificate request and module-id()
verify certificate request()
check validity of module-id()
create long term certificate()
send long term certificate()
store long term certificate()

Figure 2.12: Sequence diagram of long-term certificate request

2. In the second stage, the ITS station requests the long-term certificate from the LTCA. In order to do this, the IDM creates, with help of the HSM, an ECDSA key pair. The HSM stores the keys in a secure way and provides to the IDM only a private key handle, the public key and a signature of the public key that is signed by the device identity key. The signature of the public key is used to verify that the long-term key pair is generated by a registered HSM. It should be precluded that an attacker is able request certificates from the CA that are generated by unregistered security subsystems or by malware that is installed on the ITS station [35]. In the next step, the long-term certificate request is created using formats specified by IEEE 1609.2 or ETSI TS 103 097 and signed with the IDK and encrypted with the public key of the LTCA. Afterwards, the long-term certificate request is sent to the LTCA. When the IDM receives the response from the PKI, the long-term certificate is extracted and stored with its private key handle persistently.

In contrast to the bootstrapping process described in the first stage, the certificate request in stage two can be transmitted to the LTCA using arbitrary communication links (e.g. ITS-G5, IMT public or public WLAN). All messages that are transmitted between the requesting ITS station and the CA are encrypted with methods from IEEE 1609.2 or ETSI TS 103 097. Furthermore, the certificate request is independent from the communication protocol such as Ethernet, IP or TCP.

#### 2.5.1.3 Pseudonym Certificate Refill

The request of a pseudonym certificate (PC) is only possible if the VSS is already equipped with a valid long-term certificate. The process for requesting pseudonym certificates is comparable with the request of the long-term certificate. The sequence diagram in Figure 2.13 provides an overview of general steps.

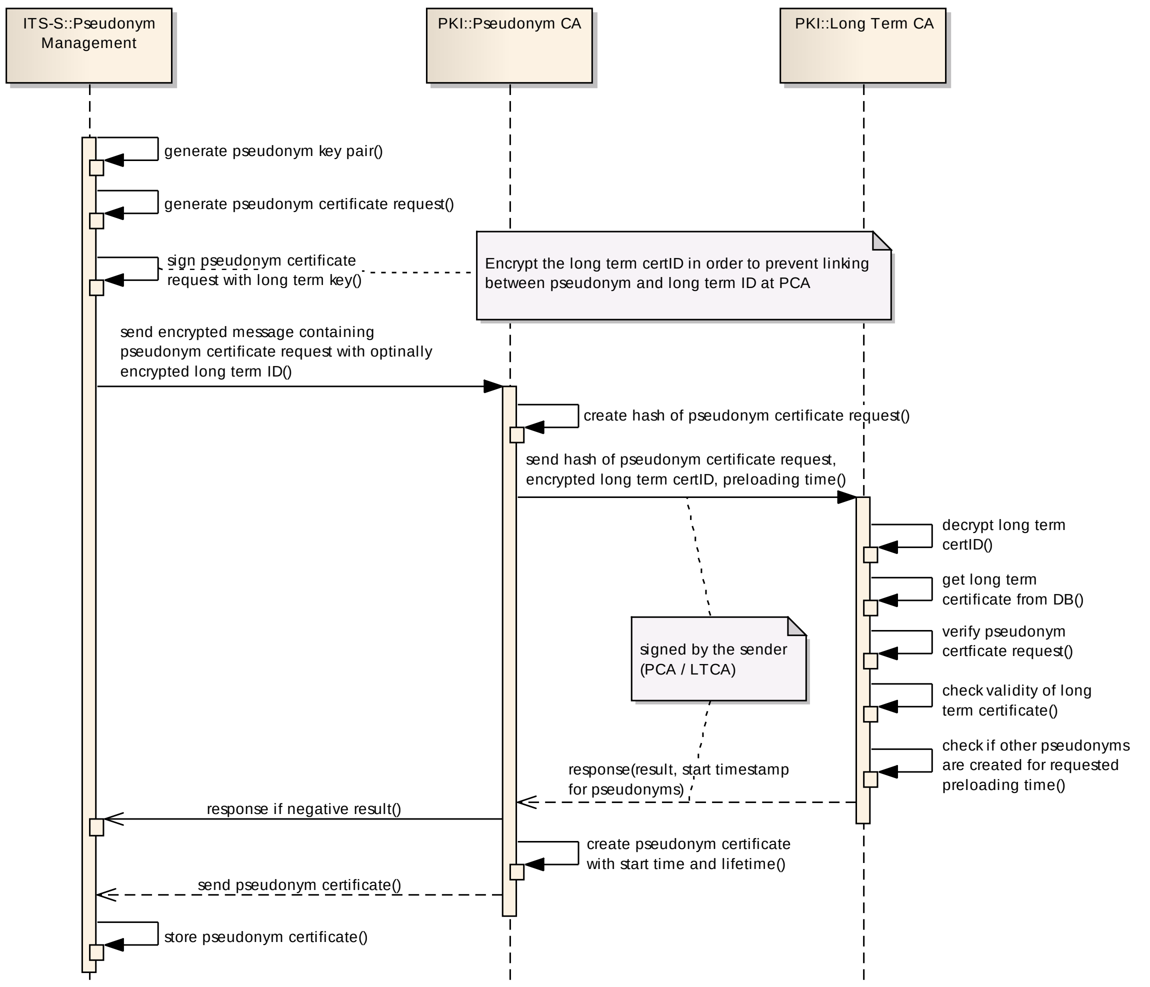

ITS-S::Pseudonym Management
PKI::Pseudonym CA
PKI::Long Term CA
generate pseudonym key pair()
generate pseudonym certificate request()
sign pseudonym certificate request with long term key()
Encrypt the long term certID in order to prevent linking between pseudonym and long term ID at PCA
send encrypted message containing pseudonym certificate request with optinally encrypted long term ID()
create hash of pseudonym certificate request()
send hash of pseudonym certificate request, encrypted long term certID, preloading time()
decrypt long term certID()
get long term certificate from DB()
verify pseudonym certficate request()
check validity of long term certificate()
check if other pseudonyms are created for requested preloading time()
signed by the sender (PCA / LTCA)
response(result, start timestamp for pseudonyms)
response if negative result()
create pseudonym certificate with start time and lifetime()
send pseudonym certificate()
store pseudonym certificate()

Figure 2.13: Sequence diagram of pseudonym certificate request

At first, the PMM triggers the generation of a new ECDSA key pair at the HSM. In order to preclude attackers from requesting PCs from the PCA the public key of the pseudonym shall be signed with the device identity private key. Then a pseudonym certificate request message is generated by using IEEE 1609.2 or ETSI TS 103 097 formats. Before the request is sent to the PCA, it is signed with the long-term private key and encrypted with the PCA public key. If the PCA should not be able to link the issued PC to its requester, i.e. to its long-term ID, the PMM has to encrypt the long-term ID with the public key of the LTCA before adding it to the request. Nevertheless, the resolution of the pseudonym owner may be necessary in FOTs due to evaluation requirements as discussed in Section 3.3. Therefore, the long-term ID may be added as readable information for the PCA. Then the request can be sent in an encrypted packet via arbitrary channels and communication protocols to the PCA. When the PMM receives the response from the PCA, the pseudonym certificate is extracted and stored with its private key handle persistently.

### 2.5.2 Security Entities Management

The Entity Authentication Module (EAM) provides various authentication services including single sign-on and sign-out. The EAM uses authentication plugins to implement the authentication process. For a more detailed description, cf. [40, Section 4.4.3].

The EAM is connected to other EVITA modules of the PRESERVE VSS as well as to the Security Event Processor (SEP) and the Cryptographic Services (CRS).

**Data Flow** Figure 2.14 shows examples of the data flow between the EAM and connected components, which are briefly explained in the following listing.

- The CCM requests entity identification and authentication of the communication end points on opening a communication channel.
- On startup, the PDM fills the list of authentication artifacts of the EAM. The PDM is also able to update the list of authentication artifacts later.
- Authentication events are logged at the SEP.
- In order to create and verify authentication tickets the respective EAM plugin uses the CRS library.

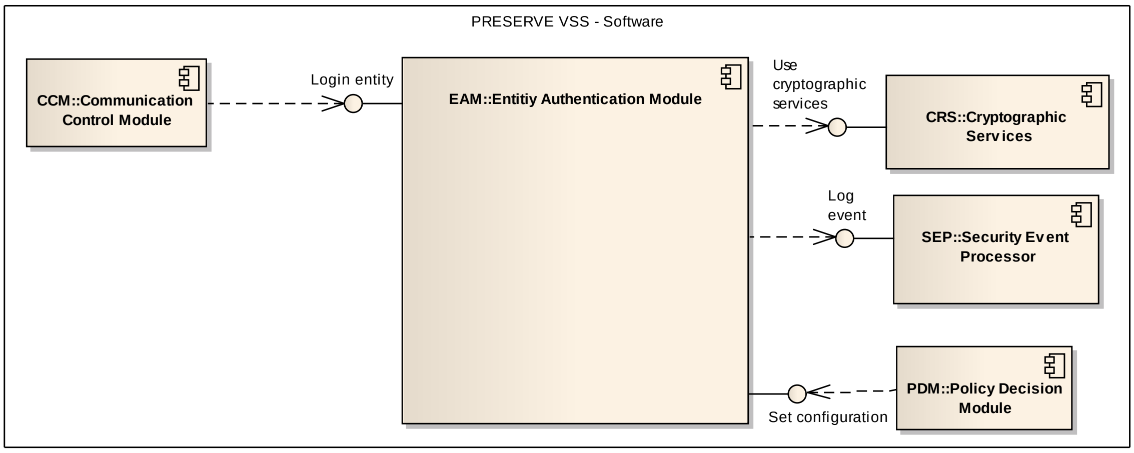

PRESERVE VSS - Software
CCM::Communication Control Module
Login entity
EAM::Entitiy Authentication Module
Use cryptographic services
CRS::Cryptographic Services
Log event
SEP::Security Event Processor
Set configuration
PDM::Policy Decision Module

Figure 2.14: Integration of the EAM into the on-board security architecture

**Dependencies** The EAM depends strongly on the following modules:

- Policy Decision Module
- Cryptographic Services

**Interfaces Between EVITA Modules** The EAM provides the following methods:

- Functions to create, log-in and log-off entities
- A function to verify authentication tickets
- A function to add authentication artifacts
- A function to log authentication events

## 2.6 Security Policies

The security policies subsystem prescribes the rules for basic tasks in a given environment. It defines for example authorization with Access Control List (ACL) rules specifying which applications or components of ITS station is granted access to system resources. According to results provided by the security analysis component discussed in Section 2.7 and the logging data, security policies can be updated.

With this respect, the PRESERVE architecture is based on components of the EVITA project. The Policy Decision Module (PDM) makes decisions regarding access control based on a local or distributed policy database. Other modules can implement a Policy Enforcement Point (PEP) that queries the PDM when access to a resource is requested. A PEP can be a implemented as a forward-PEP forwarding all requests to the PDM, but it can also be configured by the PDM with local policies to take own decisions. Further features of the PDM include configuration of more specialized PDMs (e.g., firewalls) by security configuration tickets and interoperation with other policy based access control modules such as XACML by transformation plugins.

The PDM is mainly connected to other EVITA components on the PRESERVE VSS. However, it can be connected to any security component that implements a Policy Enforcement Point, which is done by most EVITA components.

**Data Flow** Most of the following examples of the data flow between the PDM and other components have already been described in other sections about the connection of the PRESERVE modules. Hence the following listing contains only a very brief description of the examples of Figure 2.15.

- A Policy Enforcement Point, located on any security component, can forward requests to the PDM. The PDM then decides based on its policy and returns the result to the Policy Enforcement Point (PEP)
- Communication filter plugins of the CCM are configured by the PDM using a security configuration ticket.
- On opening a secure channel, the CCM requests an authorization decision by the PDM.
- The authentication artifacts of the EAM are configured by the PDM.

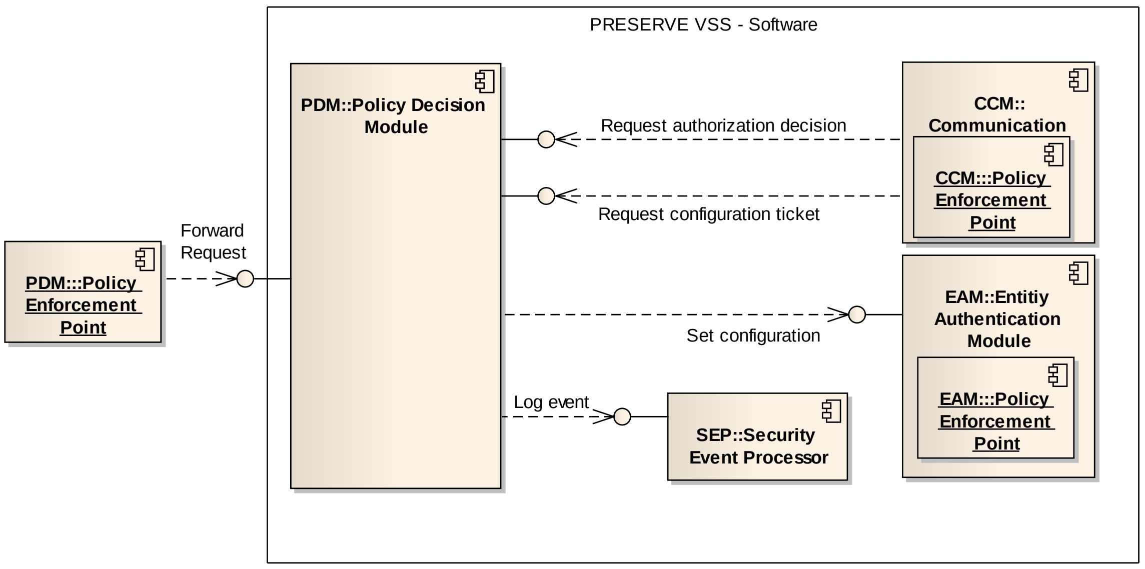

PRESERVE VSS - Software
PDM::Policy Decision Module
Request authorization decision
Request configuration ticket
CCM:: Communication
CCM:::Policy Enforcement Point
Forward Request
PDM:::Policy Enforcement Point
Set configuration
EAM::Entitiy Authentication Module
EAM:::Policy Enforcement Point
Log event
SEP::Security Event Processor

Figure 2.15: Integration of the PDM into the PRESERVE on-board security architecture

- Events about intrusion or misbehavior detection can be sent to the Security Event Processor (SEP).

A Policy Enforcement Point can be implemented by any other security component as mentioned previously and must provide the following methods

- Request an authorization decision based on the installed policies. Further authorizations can be provided by an authorization ticket.
- Requests a trust statement about an entity
- Install a security policy

## 2.7 Security Analysis

The security analysis subsystem consists of logging information related to ITS station system security in order to audit and monitor the system. Its main objective is to verify the status of the security system and to evaluate results of security policies.

The PRESERVE architecture applies a Security Event Processor (SEP) that provides mechanisms for intrusion detection, firewall tasks and intrusion response mechanisms. It includes also functionality for performing plausibility checks on incoming V2X messages. The basic module concept has been taken over from the EVITA project [40, Section 4.4.6]. The different tasks of SEP and the integration of the module into the overall VSA are described in the following. Figure 2.16 illustrates this integration.

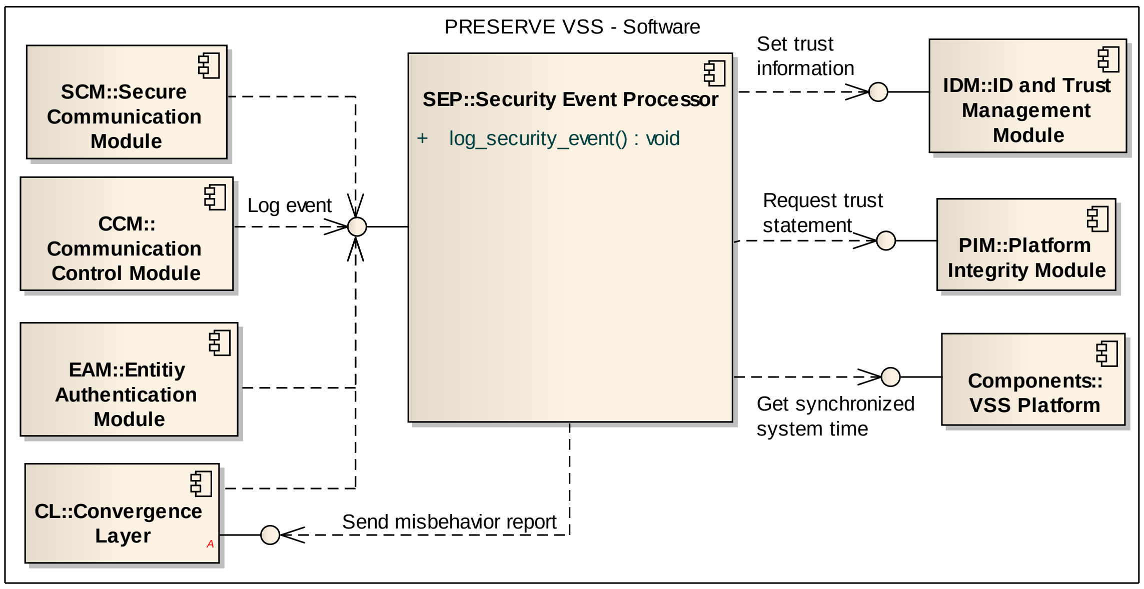

PRESERVE VSS - Software
SCM::Secure Communication Module
CCM:: Communication Control Module
EAM::Entitiy Authentication Module
CL::Convergence Layer
SEP::Security Event Processor
+ log_security_event() : void
Log event
Set trust information
IDM::ID and Trust Management Module
Request trust statement
PIM::Platform Integrity Module
Get synchronized system time
Components:: VSS Platform
Send misbehavior report

Figure 2.16: Security Event Processor

**Connection to other Modules of the PRESERVE Architecture** As shown in the component diagram in Figure 2.16 the module is connected with VSS modules from the V2X communication security area and the on-board security area.

- Communication Control Module (CCM)
- Entity Authentication Module (EAM)
- Secure Communication Module (SCM)
- ID & Trust Management Module (IDM)
- Platform Integrity Module (PIM)
- Convergence Layer

## Tasks

- The EVITA modules send status messages (e.g., authentication failures) to the SEP. SEP in turn, includes variable intrusion and misbehavior detection policies that can be linked to certain responses (e.g., alerts, shut downs). See EVITA D3.2 [40, Section 4.4.6].
- The SEP gathers from PIM security related information (e.g. about the integrity of installed software)
- CCM or SCM can register an listener at the SEP in order to get informed in case of intrusion detections. See EVITA D3.2 Section 4.4.6 [40].

- The Secure Communication Module forwards incoming messages to the SEP in order to check the plausibility of the message content (i.e. mobility data), cf. Section 2.4.3 for more details. The result may consider only a single message that is returned to the SCM or generates information about the sender's trustworthiness.
- The Convergence Layer can send log events from local V2X applications regarding specific misbehavior detection to the SEP. Detection of application specific misbehavior is not task of the VSS but the management, evaluation and generation of misbehavior reports could be sent to the PKI via Convergence Layer. This information could be used to identify attackers and faulty ITS stations by a central revocation authority.

**Timing** The plausibility checker of SEP can be used for every incoming message by the Secure Communication Module. Therefore high access frequency must be assumed and low latency requirements are given as defined in the requirements analysis of PRESERVE deliverable D1.1 [36, Section 3.1.8].

## 2.8 Secure Communication

In this section two types of secure communications are considered. In Section 2.8.1 the protection of external V2X communications is considered and in Section 2.8.2 the protection of internal communications is discussed.

### 2.8.1 External Communication

In this section the different modules are described that are responsible for the external secured V2X communications. All V2X messages except those related to the PKI are managed by the Convergence layer (CL) and the Secure Communication Module (SCM) that is detailed in the following.

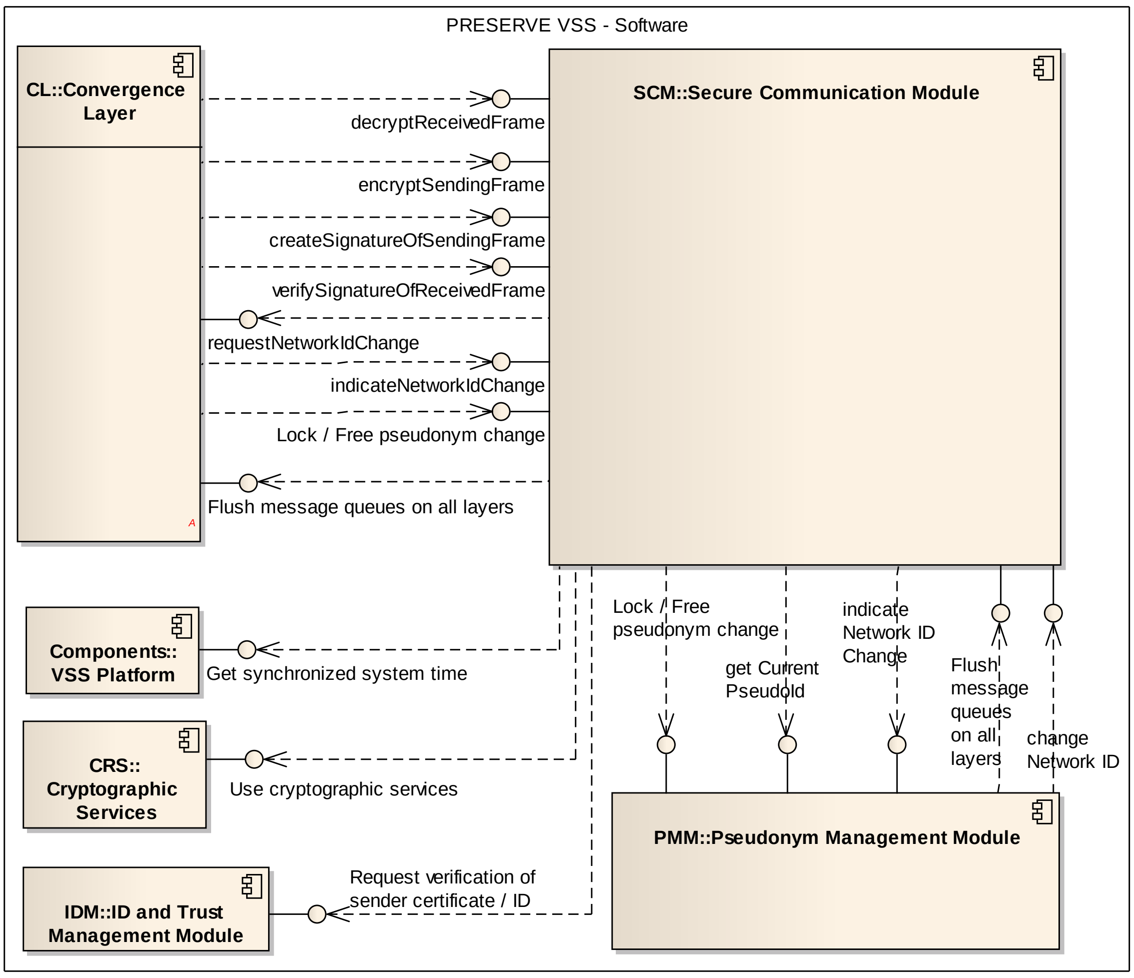
The Secure Communication Module (SCM) is the entry point of the security communication stack. As shown in Figure 2.17 the SCM is a central software component of the on-board security communication. It acts as an interface between the security components (e.g. Pseudonym Management Module, PMM) and the other components of the VSS. As the SCM is on the critical path of incoming and outgoing messages, low latency requirements are given and high access frequency by the convergence layer can be assumed as defined in the requirements analysis of PRESERVE [36, Section 3.1.8]. The Secure Communication Module implements IEEE 1609.2 v2 and ETSI TS 103 097 standards including ETSI CAM / DENM security profiles.

In the following, the internal connection points with other VSS modules are listed and example data flow descriptions are shown. Furthermore, internal dependencies to connected modules are described.

## Connection to Other PRESERVE Architecture Components

- Convergence Layer (CL)
- Cryptographic Services (CRS)
- Security Event Processor (SEP)
- Pseudonym Manager Module (PMM)
- ID & Trust Management Module (IDM)

**Data Flow** Incoming messages from the Convergence Layer are passed to the Secure Communication Module either for verification of the signature or the decryption of the payload. These operations are executed by the Cryptographic Services. At first the sender's certificate has to be verified with the interface provided by the ID & Trust Management Module.



PRESERVE VSS - Software
CL::Convergence Layer
SCM::Secure Communication Module
decryptReceivedFrame
encryptSendingFrame
createSignatureOfSendingFrame
verifySignatureOfReceivedFrame
requestNetworkIdChange
indicateNetworkIdChange
Lock / Free pseudonym change
Flush message queues on all layers
Components:: VSS Platform
Get synchronized system time
CRS:: Cryptographic Services
Use cryptographic services
IDM::ID and Trust Management Module
Request verification of sender certificate / ID
Lock / Free pseudonym change
get Current PseudoId
indicate Network ID Change
Flush message queues on all layers
change Network ID
PMM::Pseudonym Management Module

Figure 2.17: Integration of the SCM into the on-board security architecture

- If the sender (V2X network neighbor) has sent the digest of its pseudonym certificate which is unknown by the ITS station, the process stops with an error.
- If the sender has sent its full certificate chain or single pseudonym certificate, these certificates are verified. In case of success, the certificate is stored otherwise the process stops with an error.
- If the sender's digest or full certificate is known by SCM, the verification of the certificate is skipped.

When the certificate has been successfully verified, the message itself has to be verified with the public key contained in the sender's pseudonym certificate stored in the VSS. In order to check the plausibility of incoming messages the mobility data (i.e. absolute position with latitude, longitude, heading, speed, and timestamp) may be extracted from the message content and transmitted afterwards to the interface of the Security Event Processor. The result of the message plausibility as well as the reputation of the sender is returned and can be used finally by V2X applications in order to make appropriate decisions.

Outgoing messages from the Convergence Layer are passed to the Secure Communication Module (SCM) for being either signed or encrypted. If a certificate has been declared as missing during processing an incoming message and the ETSI TS 103 097 standard is applied, a request for the missing certificate(s) is added to the next outgoing message, cf. Section 2.5.1.1. The pseudonym to use is requested by the SCM from the PMM and the cryptographic operations are executed by the CRS module.

**Dependencies** The SCM strongly depends on the following modules of the PRESERVE VSA.

- Pseudonym Management Module in order to get the currently active pseudonym certificate and ID of the private key
- Cryptographic Services in order to sign / verify messages as well as encrypt / decrypt messages
- Convergence Layer in order to inform the communication stack that the IDs must be change
- ID & Trust Management Module in order to verify that the sender's certificate is known by the ITS station and in order to send pseudonym request messages to the PKI when the number of valid pseudonym certificates becomes too low
- Secure Event Processor in order to check the mobility data plausibility of incoming messages.
- A synchronized time is required in order to check for expiry of incoming certificates.

### 2.8.1.1 Outgoing Message Processing

The processing of outgoing V2X messages is one of the main tasks where security measurements are involved. In general, the PRESERVE VSS supports two types of message processing strategies as also discussed in Section 2.9.1.3.

1. The network layer can use the API of the convergence layer by calling the responsible methods in order to protect the outgoing packet appropriately. In this option, the network layer may have more control what elements of the packet are affected but also the effort may be higher at the network layer for processing the packet. This *stack controlled* usage of API access is discussed in the following paragraphs in more detail.
2. Alternatively, the network layer can simply hand-over the packet to the CL. In this case, the VSS is handling and deciding all necessary steps to protect outgoing messages. Further details regarding the *VSS controlled* packet processing can be found in the following paragraphs.

**Stack Controlled Outgoing Message Processing** Using the interfaces of the stack controlled approach on the network layer implies that two steps have to be considered by the requester.

The first step, as shown in Figure 2.18, is necessary in order to enable the optional encryption of the payload. Nevertheless, most V2X messages will not be encrypted and therefore the payload is only encapsulated into a security message format. If the network layer is storing the information whether the payload is encrypted or not then this first step can be skipped. As long as the first step is considered, the network layer uses the interface of the convergence layer to encapsulate or encrypt the payload. In case of encryption, the target MAC address from the meta data is used to get the target's certificate and its public key. In order to encrypt the payload a new symmetric key is generated and used for encrypting the data with AES_CCM. Afterwards the symmetric key is encrypted with the public key of the target using ECIES. Finally, the encrypted payload and the recipient information is returned to the network layer.

In the second step, the security header is created by the VSS as depicted in the sequence diagram of Figure 2.19. This security header contains mainly the signature and the certificate of the sender. As discussed in Chapter 1, primarily single-hop broadcast message transmission is considered in the VSA. Nevertheless, multi-hop message transmission should be supported in general by applying end-to-end message protection. This means, the receiver can only verify the originator of the message and not the intermediate ITS stations that have forwarded the message. Multi-hop message protection mechanisms will be considered in future works. In order to protect originator message content, at first the network layer sets all mutual fields of the network header to zero. Then the packet data, consisting of network header and the optionally encapsulated payload, as well as the meta data and encoding flags are submitted to the VSS using the interface of the convergence layer. Inside the VSS at first the active pseudonym is checked whether it supports the required permissions. If this is not the case, a pseudonym change has to be triggered

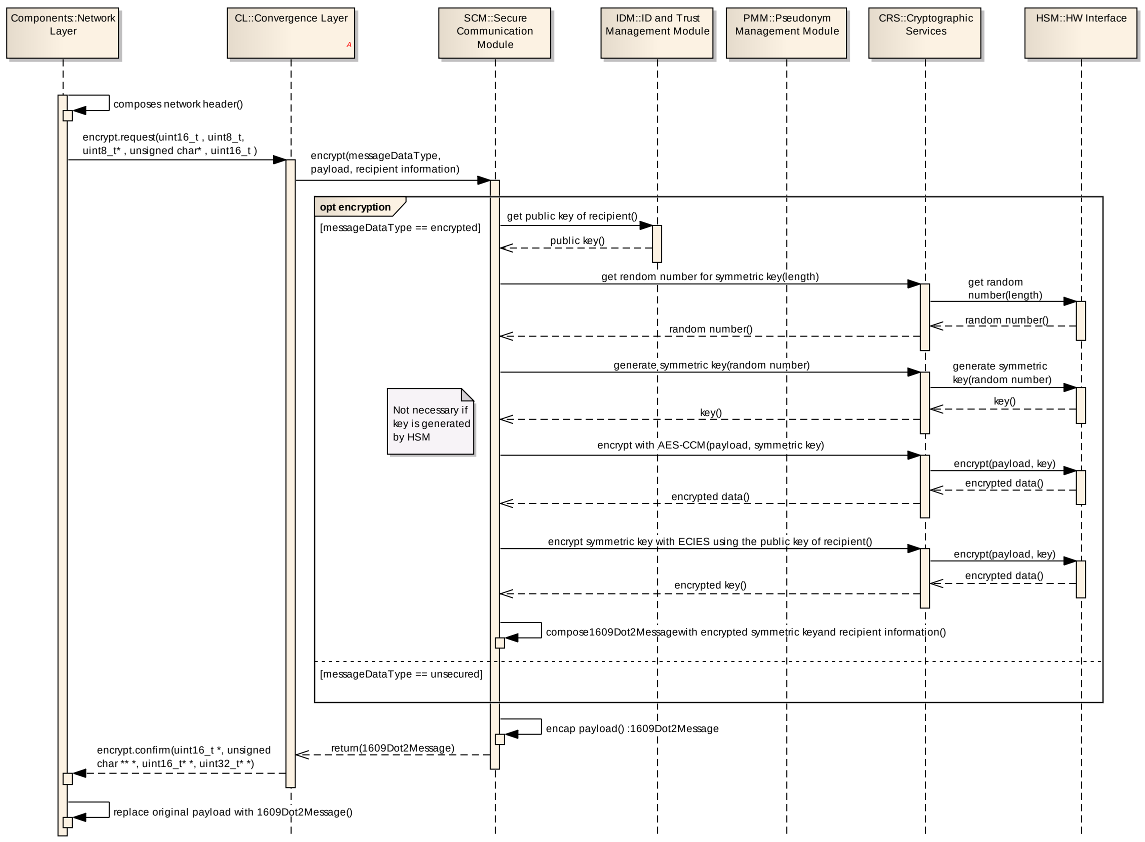

Components::Network Layer
CL::Convergence Layer
SCM::Secure Communication Module
IDM::ID and Trust Management Module
PMM::Pseudonym Management Module
CRS::Cryptographic Services
HSM::HW Interface
composes network header()
encrypt.request(uint16_t , uint8_t, uint8_t* , unsigned char* , uint16_t )
encrypt(messageDataType, payload, recipient information)
opt encryption
[messageDataType == encrypted]
get public key of recipient()
public key()
get rendom number for symmetric key(length)
get random number(length)
random number()
random number()
generate symmetric key(random number)
generate symmetric key(random number)
key()
key()
Not necessary if key is generated by HSM
encrypt with AES-CCM(payload, symmetric key)
encrypt(payload, key)
encrypted data()
encrypted data()
encrypt symmetric key with ECIES using the public key of recipient()
encrypt(payload, key)
encrypted data()
encrypted key()
compose1609Dot2Messagewith encrypted symmetric keyand recipient information()
[messageDataType == unsecured]
encap payload() :1609Dot2Message
return(1609Dot2Message)
encrypt.confirm(uint16_t *, unsigned char ** *, uint16_t* *, uint32_t* *)
replace original payload with 1609Dot2Message()

Figure 2.18: Stack controlled outgoing message processing (step 1)

as described in more details in Section 2.4.4.3. Then the given encoding flags are evaluated and the appropriate security profile is selected for outgoing messages. The PRESERVE VSS supports three different profiles, namely IEEE 1609.2 [22, annex B], ETSI TS 102 867[11, Section 6.2], and ETSI TS 103 097 [14]. Afterwards, the *SignedMessage* format is created that contains the certificate as *SignerIdentifier*, a *ToBeSignedMessage* and the signature. As only a security header should be generated in this step, the *ToBeSignedMessage* uses the *ContentType* "signed_external_payload" that means the packet data is not part of the *SignedMessage* structure. Finally, the *SignedMessage* is encapsulated and returned to the network layer as security header. Before giving the packet to the access layer, the network layer adds the security header and sets the mutable fields.

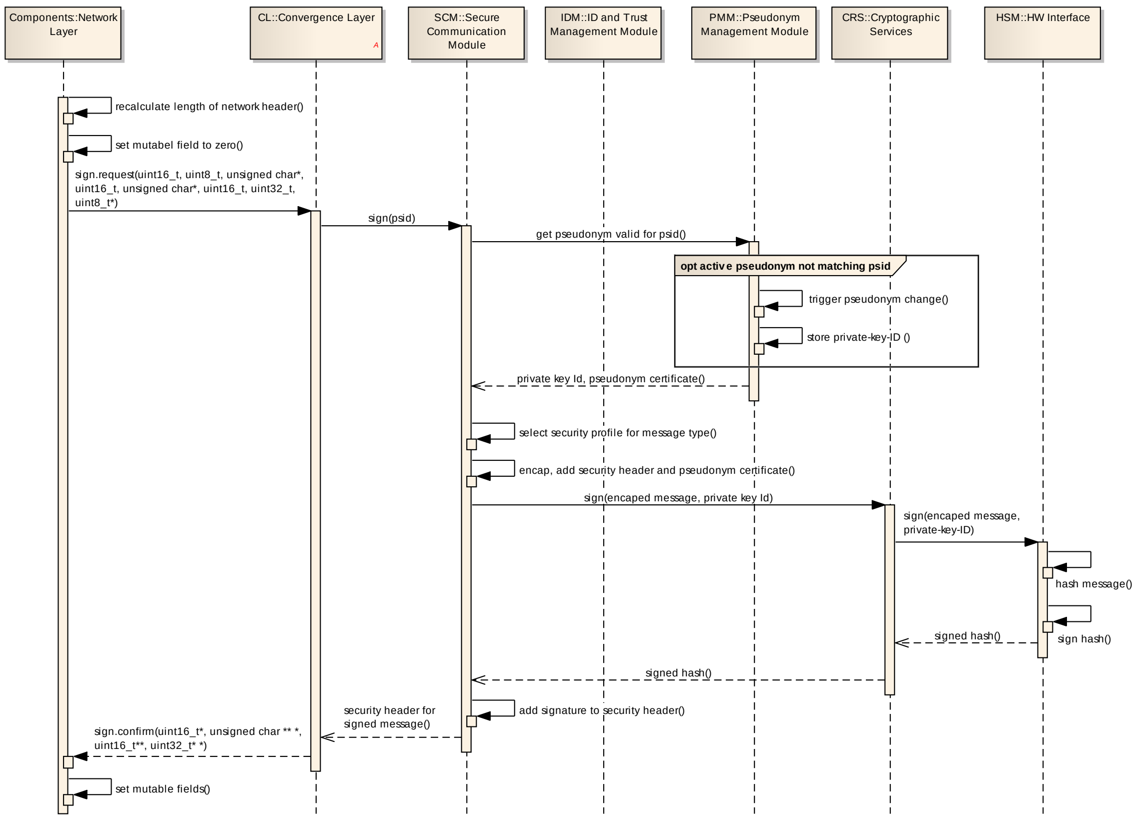

Components::Network Layer
CL::Convergence Layer
SCM::Secure Communication Module
IDM::ID and Trust Management Module
PMM::Pseudonym Management Module
CRS::Cryptographic Services
HSM::HW Interface
recalculate length of network header()
set mutabel field to zero()
sign.request(uint16_t, uint8_t, unsigned char*, uint16_t, unsigned char*, uint16_t, uint32_t, uint8_t*)
sign(psid)
get pseudonym valid for psid()
opt active pseudonym not matching psid
trigger pseudonym change()
store private-key-ID ()
private key Id, pseudonym certificate()
select security profile for message type()
encap, add security header and pseudonym certificate()
sign(encaped message, private key Id)
sign(encaped message, private-key-ID)
hash message()
sign hash()
signed hash()
signed hash()
add signature to security header()
security header for signed message()
sign.confirm(uint16_t*, unsigned char ** *, uint16_t**, uint32_t* *)
set mutable fields()

Figure 2.19: Stack controlled outgoing message processing (step 2)

**VSS Controlled Outgoing Message Processing** The VSS controlled processing of outgoing V2X messages is described in this paragraph. As depicted in Figure 2.20 the network layer can use the VSS controlled interface of the PRESERVE API in contrast to the stack controlled interfaces. In this case the convergence layer of the VSS is responsible for the appropriate message processing regarding the exchange of payload in case of encryption and the generation of the security header and its integration into the packet. As shown in the sequence diagram in Figure 2.20 the convergence layer has to parse at first the packet and exchange afterwards the payload. Then the recalculation of network header length fields have to be done and mutable fields of the network header have to be set to zero before the security header can be created with the same internal processes as presented in the stack controlled message processing strategy. Finally, the mutable fields have to be reseted and the security header is added to the packet.

This VSS controlled message processing strategy has the advantage that security operations can be adapted over time without adaption of the communication stack. On the other hand, the structure of the packet including all headers has to be known by the convergence layer. Changes and adaptations of the packet structure may entail also an update of the convergence layer.

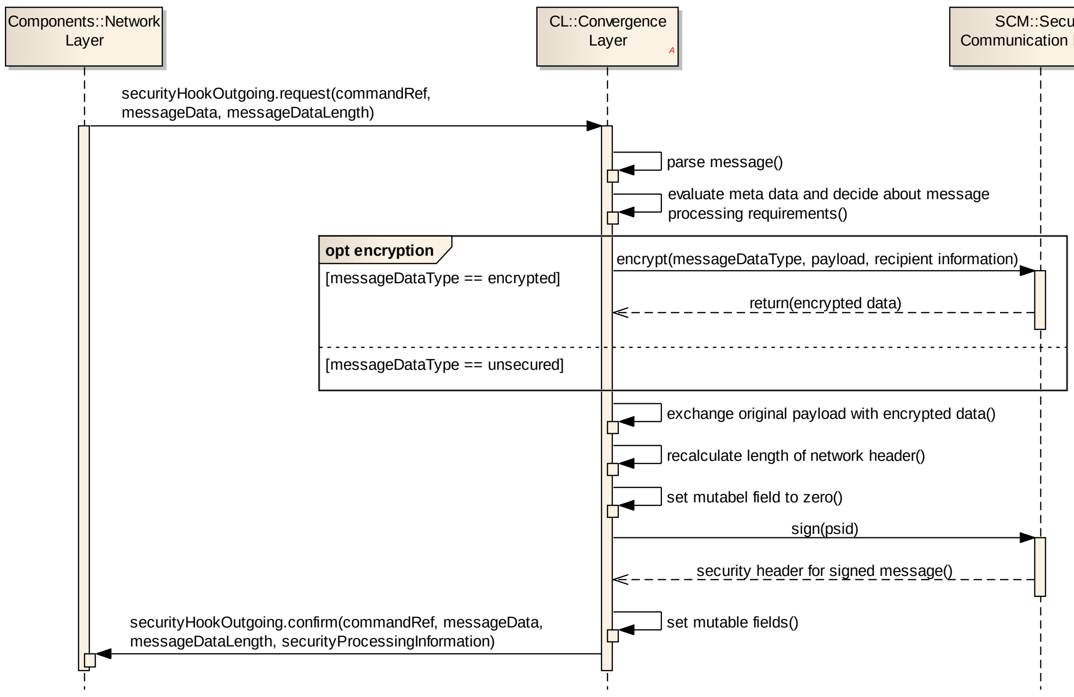

Components::Network Layer
CL::Convergence Layer
SCM::Secure Communication Module
securityHookOutgoing.request(commandRef, messageData, messageDataLength)
parse message()
evaluate meta data and decide about message processing requirements()
opt encryption
[messageDataType == encrypted]
encrypt(messageDataType, payload, recipient information)
return(encrypted data)
[messageDataType == unsecured]
exchange original payload with encrypted data()
recalculate length of network header()
set mutabel field to zero()
sign(psid)
security header for signed message()
set mutable fields()
securityHookOutgoing.confirm(commandRef, messageData, messageDataLength, securityProcessingInformation)

Figure 2.20: VSS controlled outgoing message processing

#### 2.8.1.2 Incoming Message Processing

Similar to the outgoing message processing, the VSA provides for incoming message processing also supports two strategies that can be used by the network layer. Using the stack controlled process, the network layer is responsible for calling the responsible methods of the convergence layer API in order to verify and decrypt incoming messages. If the VSS controlled strategy is used, the convergence layer is responsible for all security operations and the network layer has to provide only the incoming packet. Both strategies are described in more detail in the following paragraphs.

**Stack Controlled Incoming Message Processing** When a message is received, then the network layer has to set at first all mutable fields to zero, as shown in Figure 2.21. Then the security header as well as the external message data, consisting of the network header and payload, is given to the VSS for verification. Applying the stack controlled strategy, the interfaces of the convergence layer are used. Inside the VSS, at first the certificate is extracted from the security header and checked whether this sender is already known. As long as the sender is known and trusted, it is not necessary that the certificate of the sender is verified again. If different Pseudonym CAs are used it may be necessary that also the certificate of the pseudonym certificate issuer has to be verified if a message is received from an unknown sender. In the second step, the message data is verified by using the public key from the received certificate. Afterwards, the message plausibility is checked by SEP. Finally, the result is returned to the network layer in form of meta data

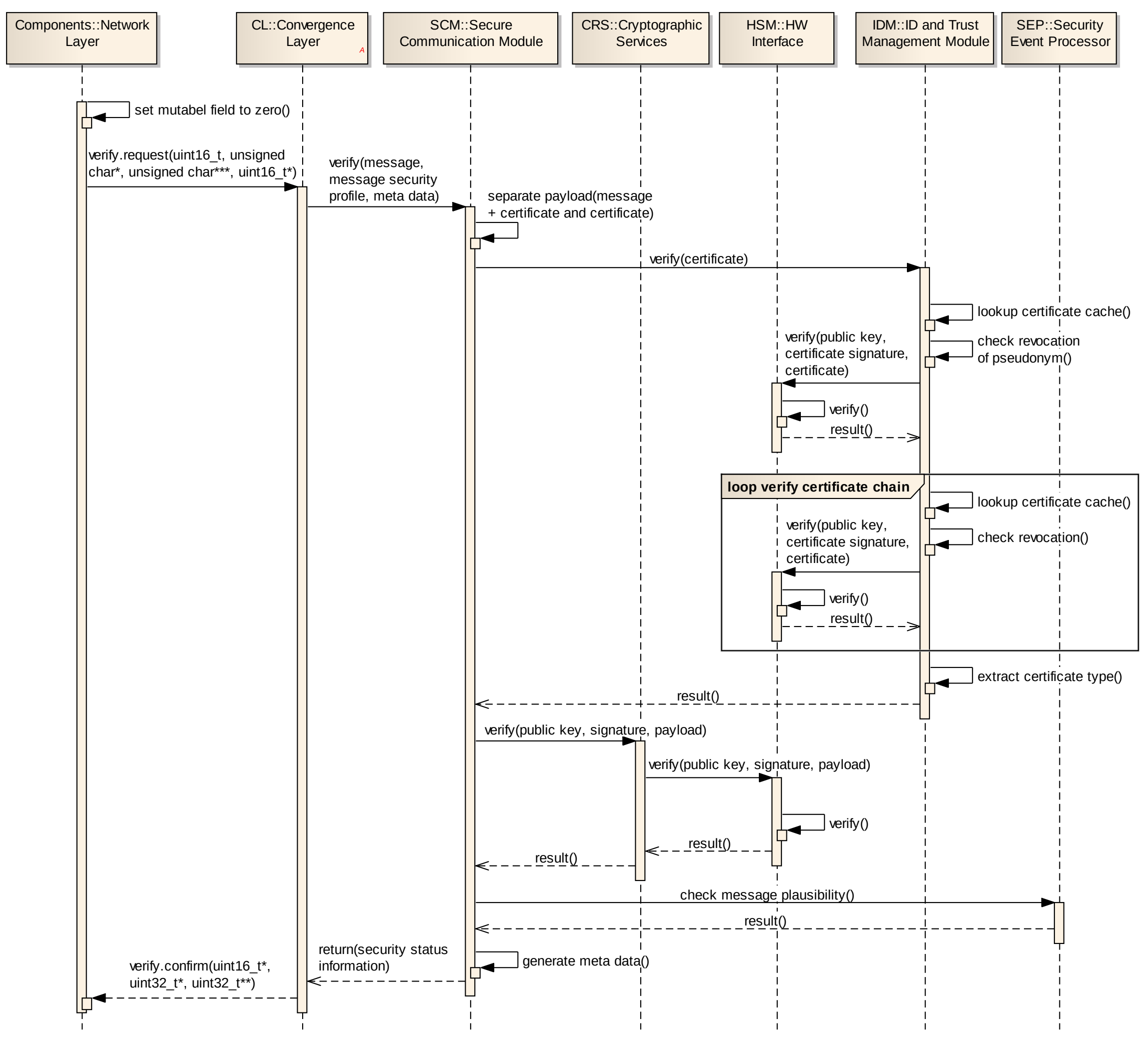

Components::Network Layer
CL::Convergence Layer
SCM::Secure Communication Module
CRS::Cryptographic Services
HSM::HW Interface
IDM::ID and Trust Management Module
SEP::Security Event Processor
set mutabel field to zero()
verify.request(uint16_t, unsigned char*, unsigned char***, uint16_t*)
verify(message, message security profile, meta data)
separate payload(message + certificate and certificate)
verify(certificate)
lookup certificate cache()
check revocation of pseudonym()
verify(public key, certificate signature, certificate)
verify()
result()
loop verify certificate chain
lookup certificate cache()
check revocation()
verify(public key, certificate signature, certificate)
verify()
result()
extract certificate type()
result()
verify(public key, signature, payload)
verify(public key, signature, payload)
verify()
result()
result()
check message plausibility()
result()
return(security status information)
generate meta data()
verify.confirm(uint16_t*, uint32_t*, uint32_t**)

Figure 2.21: Stack controlled incoming message processing (step 1)

that is generated by the convergence layer. This meta data may contain the processing result of cryptographic certificate and message verification, permission information that are available in the received pseudonym certificate and plausibility information. This information can be used subsequently on application layer in order to check whether the message can be used.

Before the verified message can be given to the communication stack on network layer, the encapsulated payload has to be extracted as shown in Figure 2.22. This step can be skipped if the network layer has noticed in its network header whether the payload is encrypted or encapsulated. The network layer extracts the data from the packet and uses the method of the PRESERVE API. Inside the VSS the message structure is read and according to the *ContentType* the contained data is decrypted using ECIES or simply extracted and returned to the network layer as clear text. The network layer is able to

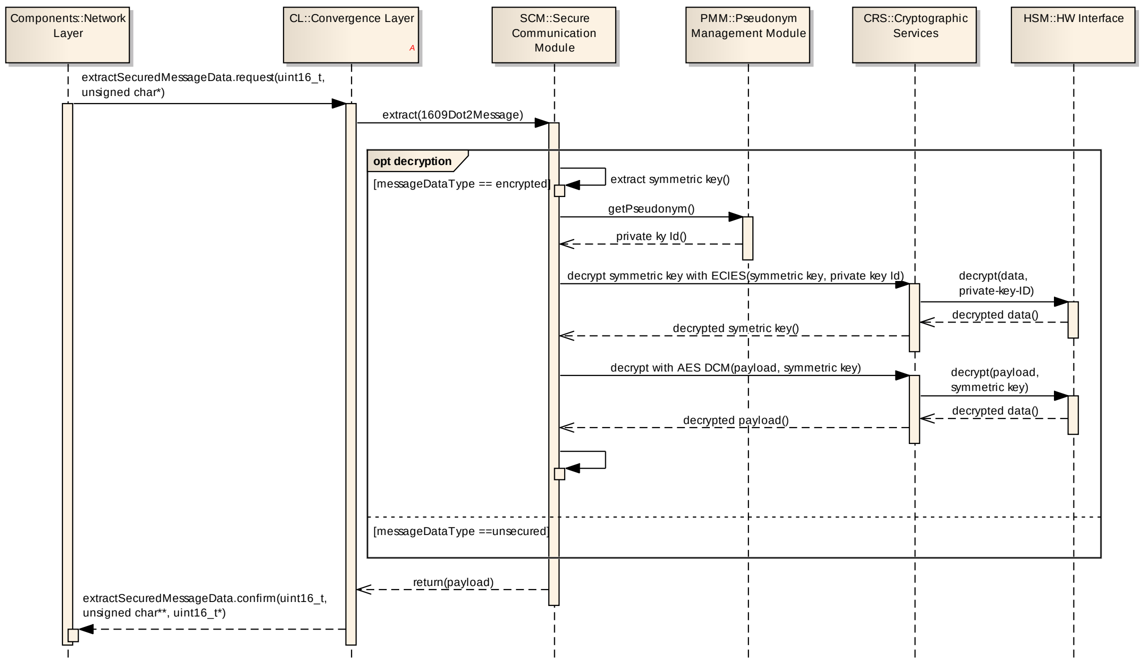

Components::Network Layer
CL::Convergence Layer
SCM::Secure Communication Module
PMM::Pseudonym Management Module
CRS::Cryptographic Services
HSM::HW Interface
extractSecuredMessageData.request(uint16_t, unsigned char*)
extract(1609Dot2Message)
opt decryption
[messageDataType == encrypted]
extract symmetric key()
getPseudonym()
private ky Id()
decrypt symmetric key with ECIES(symmetric key, private key Id)
decrypt(data, private-key-ID)
decrypted data()
decrypted symetric key()
decrypt with AES DCM(payload, symmetric key)
decrypt(payload, symmetric key)
decrypted data()
decrypted payload()
[messageDataType ==unsecured]
return(payload)
extractSecuredMessageData.confirm(uint16_t, unsigned char**, uint16_t*)

Figure 2.22: Stack controlled incoming message processing (step 2)

exchange subsequently the encapsulated data with the original payload and provide the packet to the next upper layer.

#### 2.8.1.3 VSS Controlled Incoming Message Processing

If the VSS controlled strategy is used by the network layer the hooking interfaces of the PRESERVE API can be used. Figure 2.23 shows in the sequence diagram the related process.

### 2.8.2 Internal Communication

Besides secure external V2X communication, a secure on-board network is another important part of a complete security architecture. The latter subject is covered by the EVITA project, whose objective is, as stated in the abstract of the EVITA deliverable D3.2 [40], "to design, verify, and prototype an architecture for automotive on-board networks where security-relevant components are protected against tampering and sensitive data are protected against compromise."

The on-board network to be protected consists of the PRESERVE VSS together with different kinds of on-board components such as sensors, ECUs, the head unit or internal buses like CAN, in the following referred to as on-board entities. Thus, the PRESERVE VSS as part of the on-board network needs to include the following EVITA components in order to fulfill the required objectives.

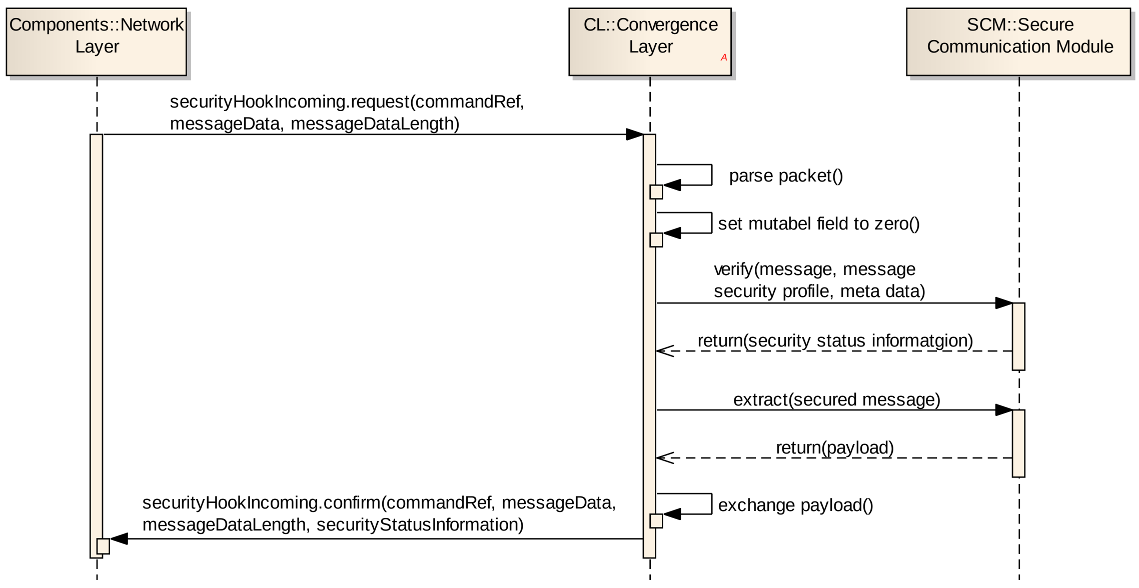

Components::Network Layer
CL::Convergence Layer
SCM::Secure Communication Module
securityHookIncoming.request(commandRef, messageData, messageDataLength)
parse packet()
set mutabel field to zero()
verify(message, message security profile, meta data)
return(security status informatgion)
extract(secured message)
return(payload)
exchange payload()
securityHookIncoming.confirm(commandRef, messageData, messageDataLength, securityStatusInformation)

Figure 2.23: VSS controlled incoming message processing

- Communication Control Module (CCM) described in this section
- Entity Authentication Module (EAM) described in Section 2.5.2
- Policy Decision Module (PDM) described in Section 2.6
- Platform Integrity Module (PIM) described in Section 2.4.2

Since the PRESERVE HSM has all functionality of an EVITA HSM, it also acts as EVITA HSM and thus makes it possible to have an EVITA compliant on-board network. However, establishing a secure on-board network is not a crucial requirement of PRESERVE in the sense that the PRESERVE VSS depends on the existence of an EVITA-secured on-board network. The PRESERVE VSS rather offers an optional possibility to do so. To what extent the on-board network will be secured depends on the equipment of the other on-board entities with EVITA functionality. The full integration of these modules is recommended when FOTs rely on the use of protected communication between on-board sensors, ECUs or camera and the VSS. This would require that the FOT uses vehicles that are equipped with sensors and ECUs extended by EVITA hardware security modules.

Based on the EVITA deliverable D3.2 [40], it follows a short overview of the EVITA components incorporated into the PRESERVE VSS. The following descriptions are taken from [40] and slightly modified. A complete description of the interfaces of the EVITA modules is not given in this document since these modules are reused from the EVITA project and are not used by any non-EVITA component of the PRESERVE VSA. For more detailed descriptions, cf. [40, Section 4.4].

The Communication Control Module (CCM) provides a high-level interface for secured communication between different on-board entities. In order to establish channels and actually send and receive data, the CCM uses protocol plugins that implement the functionality of the desired communication protocol. Besides, routing and communication filtering can be integrated by implementing a routing plugin or a communication filter plugin respectively.

The CCM is connected to the EVITA components included in the PRESERVE VSS (i.e., EAM, PDM) as well as to EVITA components on other on-board components such as sensors, ECUs or the head unit. Furthermore, it is connected to the Cryptographic Services, the Security Event Processor and the Privacy-enforcing Runtime Architecture inside the PRESERVE VSS.

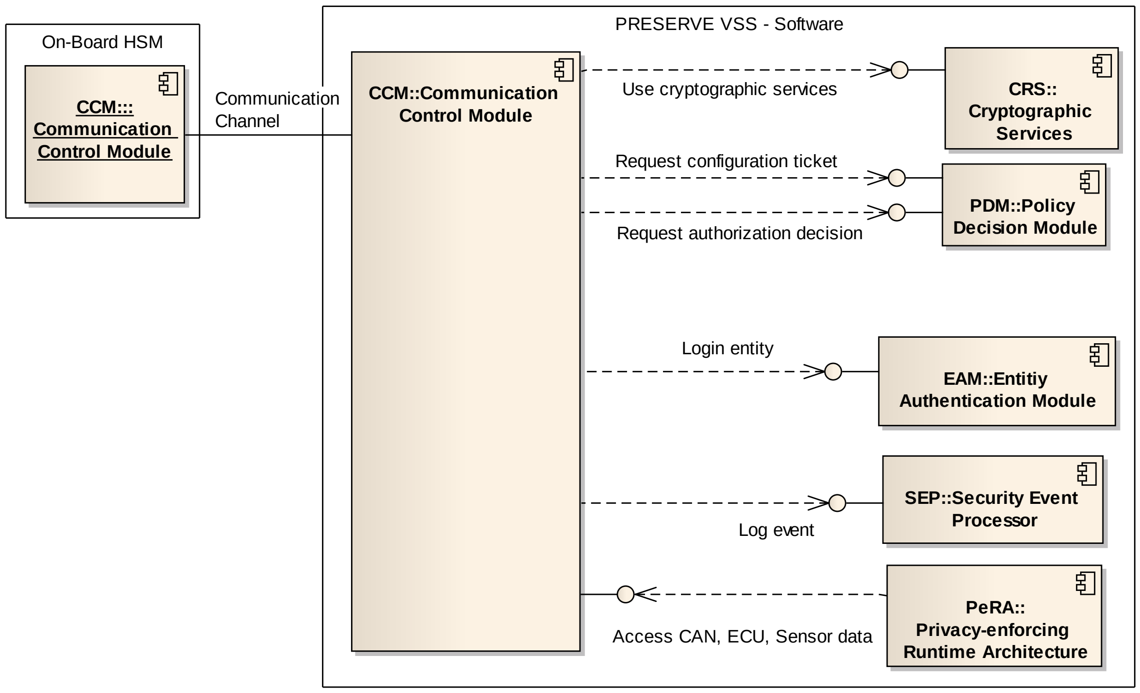

On-Board HSM
CCM::: Communication Control Module
Communication Channel
PRESERVE VSS - Software
CCM::Communication Control Module
Use cryptographic services
CRS:: Cryptographic Services
Request configuration ticket
Request authorization decision
PDM::Policy Decision Module
Login entity
EAM::Entitiy Authentication Module
Log event
SEP::Security Event Processor
Access CAN, ECU, Sensor data
PeRA:: Privacy-enforcing Runtime Architecture

Figure 2.24: Integration of the CCM into the on-board security architecture

**Data Flow** Examples of the data flow between the CCM and the connected components are shown in Figure 2.24 and briefly described in the following listing.

- All on-board entities, in the same or in other vehicles, that are equipped with a CCM are able to establish a secure communication with the PRESERVE VSS.
- The CCM uses the CRS library to encrypt/decrypt and sign/verify messages.
- When the CCM establishes a secure channel, both endpoints have to be authenticated. This is done by requesting authentication tickets from the EAM.
- Before establishing a secure channel, the CCM also has to check, whether the right to open a secure channel is granted to the involved entities. This is done by requesting an authorization decision from the PDM.

- For configuration of a communication filter, the CCM can request a security configuration ticket from the PDM.
- Events linked to secure communication like e.g., opening a channel are logged by the SEP.

**Dependencies** The CCM depends strongly on the following modules:

- Cryptographic Services
- Entity Authentication Module
- Security Event Processor

**Interfaces Used Between EVITA Modules** The CCM offers the following public methods for opening and using a secure channel for on-board communication.

- Functions for channel opening and closing
- Functions to bind and release sockets
- A function to send messages over a secure channel

If a service is to be provided to other entities, a server interface has to be implemented providing the following functions.

- A function that accepts client connections
- A function that receives data

The CCM uses the following plugins that have to be implemented for a specific platform. The Protocol plugin is necessary for the operability of the CCM, whereas the other plugins only have to be implemented, if routing or communication filtering is planned to be used.

- CCM_Adress_Resolution_Plugin
- CCM_CommunicationFilter_Plugin
- CCM_Protocol_Plugin
- CCM_Router_Plugin

## 2.9 Interfaces of the On-Board V2X Security Subsystem

The VSS of a vehicle or RSU has to be connected with different other components of the on-board system in order to provide security support for internal and external communications. In Section 2.9.1 the integration of the VSS into the V2X communication stack is discussed. Most relevant aspect considered in this section is the integration of security information into packets that are exchanged with external V2X communication entities. In Section 2.9.2 the on-board internal exchange of security related meta information is discussed. In order to perform the appropriate security operations it is relevant to exchange information between different components of a communication stack. In Section 2.9.3 the parallelism of security operations is discussed.

### 2.9.1 V2X Communication Stack

In the following section first the general V2X communication stack of the ETSI ITS-S reference architecture [5] is presented from security perspective. Subsequently, different alternatives are discussed at which position of the V2X packet the security information could be integrated. Finally, we describe and motivate the solution chosen in the PRESERVE VSA.

#### 2.9.1.1 V2X Communication Stack Overview

Many projects and standards such as CALM, C2C-CC, IEEE and ETSI have defined several ITS communication architectures. For PRESERVE, the ITS station architecture is considered that is based on the European ITS architecture described in ETSI EN 302 665 [5]. From a communication perspective an ITS station architecture based on the reference protocol stack, shown in Figure 2.25, is similar to the traditional OSI layered model. This

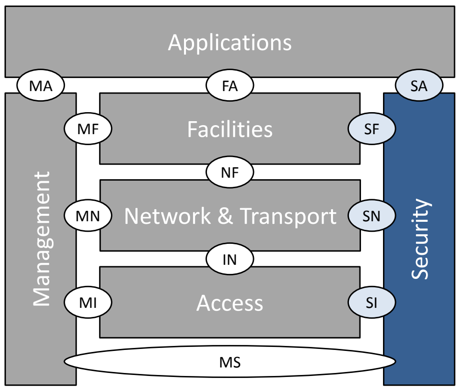


Figure 2.25: ITS-S reference architecture [5]

protocol stack consists of four horizontal layers: Access, Network & Transport, Facilities, and Applications. Other modules in the ITS station architecture are the management and security entities. The management and security entities are cross layers which provide management and security services to different layers of the communication stack. The relevant interfaces for the security layer are *SA, SF, SN, SI, MS* which are introduced in this document in Section 2.2.2. Communication layers and entities in the ETSI ITS communication architecture interacts with Security entities via SAP (Service Access Point) as defined in the OSI Reference Model, cf. [16]. The ITS communication layers are interconnected to security entity via SF, SN and SI SAPs respectively to access the security services provided by ITS Security entity. Management entity and security entity are interconnected via the MS SAP to exchange management information and applications are communicating with security entity via SA SAP, cf. [5]. Security is providing various services to applications via SA-SAP, including security management services and security services. Several aspects related to this ITS-S reference architecture and its layers are detailed in the following.

**Access Layer** The access layer has the purpose to interface with the different communication technologies that are available in an ITS station. Both wired and wireless access technologies are supported for station-internal communications and station-external communications. The access technology dedicated for safety-related applications is ITS-G5 based on the European ETSI profile of IEEE 802.11p.

**Network & Transport Layer** On top of the access layer, the network & transport layer provides the transport of data between source and destination ITS stations. Figure 2.26 shows that the networking and transport layer is divided into two parts: ITS-specific network and transport protocols and IP-based network and transport protocols. For the ITS-specific network and transport part, Basic Transport Protocol [13] and Geonetworking [12] protocols are defined as ITS transport protocol and ITS networking protocol respectively. On the other side, the IPv6 stack is composed by the IPv6 protocol and related transport protocols such as UDP and TCP. Additionally, the IP stack can run at the top of the GeoNetworking protocol (IPv6 over geonetworking).

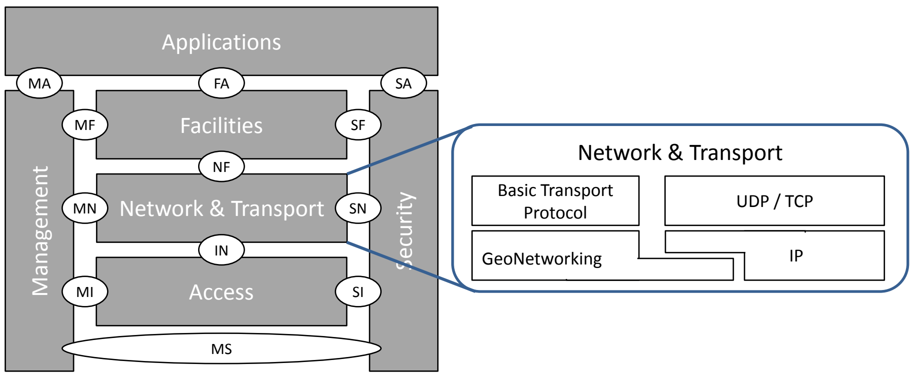


Figure 2.26: ITS-S reference architecture: Networking and Transport layer

**Facilities Layer** The facilities layer is integrated between application layer and network & transport layer and acts as a middleware as shown in Figure 2.27. This layer is composed by three function blocks: application support, information support, and communication support. These blocks contain support for both, ITS-based applications and IP-based applications.

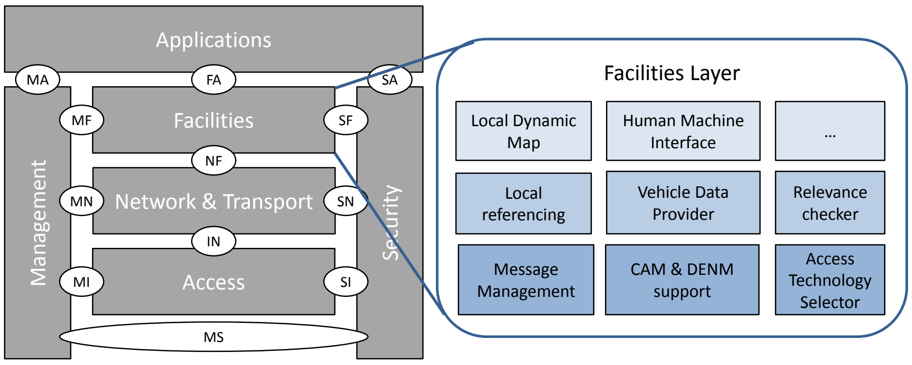


Figure 2.27: ITS-S reference architecture: Facilities layer

- Support of V2X applications is considered as the kernel of the ITS facility layer. It contains principally a human-machine interface (HMI) that presents information to the user of the system and a local dynamic map (LDM) that provides information about the ITS station environment.
- Information support contains a Vehicle Data Provider which manages the access from applications to vehicle data, a relevance checker calculating the relevance of each received message (an information filter) and others modules depending on implementation. The relevance checker may have a link to PeRA if sensor data is to be retrieved from there.
- Communication support is responsible for message transmission and is composed mainly by messaging support and traffic management message support responsible for the generation, extraction, and management of ITS-based messages (CAM and DENM) and road traffic efficiency messages respectively. The Access Technology Selector can be used to choose the radio technology for message transmission.

**Application Layer** European ITS define three groups of applications in cooperative ITS: Road Safety, Traffic Efficiency and Other Applications. Different applications can exist in parallel in an ITS station. All information, data, and communication functions that are used commonly by applications are provided by the facilities layer. Therefore, application support is regarded as the kernel of the ITS facility layer. It contains principally a human-machine interface (HMI) that presents information to the user of the system, a local dynamic map (LDM) that provides information about the ITS station environment, and a processor for Service announcement message (SAM).

#### 2.9.1.2 Position of Security Processing in the Communication Stack

The security layer is a cross-layer component that provides security services to all communication stack layers. Processing security in all communication layers will incur high overhead in terms of processing, data, and communication [29]. The security overhead can degrade performance requirements for safety applications. In order to limit security overhead, security processing should be done primarily at only one layer of the ITS communication stack. The question remain in which layer of ITS communication stack security processing should be done: at Network & Transport layer or at Facilities layer. In order to answer this question, arguments for each approach are addressed in the following.

**Arguments for Networking and Transport Layer** Applying security on networking and transport layer avoids that security depends on the correct implementation of a certain application or could be undermined by changing or modifying applications. The network layer position is more generic than the application one. Indeed, it can cover other message types without need for re-defining the security payload. Moreover, the security will be implemented on the communication unit where, for example also IPsec could benefit from the availability of a hardware security module. Data of the network header are protected by digital signature in order to avoid attacks on the routing. If the network stack would be able to pass meta data about a packet/message from the network layer to higher layers in the stack, then all cryptographic security processing could be performed on the network layer. Then the packet is tagged accordingly and this data is available to applications for decision processes (e.g. if an emergency vehicle contains this status in the certificate, this information becomes part of the meta data after verification of the certificate/signature).

In ETSI EN 302 636-4-1 [15] it is proposed that the network header consist of three parts: Basic Header, Common Header, and Extended Header. The security envelope shall include only the Common and Extended Headers, and exclude the Basic Header. As a consequence, the security processing must be performed after packet reception on network layer, before the operations for multi-hop routing are finished.

The sim$^{TD}$ project [3][38] is performing security processing at the network layer and finds that meta data processing is a strongly desirable feature, as otherwise the data verified by the security system (e.g. attributes in the certificates) gets lost between layers. PRESERVE does not consider potential security compromise between layers as a major issue. Indeed, if an attack manages to manipulate the communication stack in a way that modifications of data between layers becomes possible, this attacker will likely also be able to directly manipulate facilities or applications. Finally, this option fits with the testing plans of PRESERVE where an application unit might not be available all the time.

**Arguments for Facilities Layer** The objective of PRESERVE is to secure applicative messages such as CAM and DENM. These message types are generated in the facilities layer. A logical method is to apply the security services associated at the location of the message generation. Moreover, the security services to apply depend on the type of the message (based on the security requirements of the application that generates one or more types of messages). For each type of message, the security functions need to be redefined to necessary application. As we can not have a generic security processing for all messages and consequently for all packet, applying security on facilities layer avoids exchange of security meta data between layers of the communication stack.

In some use cases, we need to authenticate the sending application of the message. So, applying a network-layer authentication is irrelevant.

We can secure the applicative data in layers below (network) but it is not the security of a message but the security of a packet that encapsulates an applicative payload such as CAM or DENM. This requires more processing where security is applied (e.g. network layer or security layer). Indeed, this layer has to interpret the packet, limit the fields and if necessary cancel the network header fields related to routing. And thus, it leads to more data streams exchanged between the network and facilities layers in both directions.

**Discussion and PRESERVE Solution** PRESERVE discussed about whether ETSI TS 103 097 [14] or IEEE 1609.2 [22] is specified for facilities or application layer security, while it clearly seems to be above network layer. Security processing is done at a layer in-between Facilities and Network layer. The security payload is specific per message type, however, the mechanisms are generic between messages. We argue that we need to be rather flexible with our mechanisms so that they can be applied at various layers.

Having different applications with different security requirements might complicate their security processing at network layer. In this case, we have a need to identify an interface mechanism by which applications can signal the security requirements of a message or connection to the security layer.

However, in case of forwarding packets in multi-hop scenarios, security checking cannot be done at the facilities layer, as packets will be forwarded directly in the communication unit. Indeed, some data would then be unprotected at the network layer. Processing security at facilities layer has the disadvantage that network layer information cannot be protected by the same signature. Assuming that in the future more complex geonetworking protocols will be used and relevant information in the network header needs to be protected, this would lead to process security on two different layers, which would double the processing and packet overhead.

To conclude, security processing at the network layer will permit the security to be transparent for the facilities layer. For outgoing messages, the application just sets the configuration of security requirements for specific messages, e.g. integrity or confidentiality. For incoming message, the network layer performs security processing (before applications can access the data) and then creating a certain "per packet" or "per session" security state that an application can access later on. Then, it forwards the data to the facilities layer with the meta data needed. Indeed, attaching meta data permits the application to have the possibility to check the meta data from network layer (e.g. for consistency checks of location data) and security information from the respective layer. An option could also be to inform the facilities layer that the packet was not correctly signed and so leave to the facilities layer the final decision to discard it or not. With the aforementioned arguments, the PRESERVE project decides that the security processing will be done at the network layer as shown in Figure 2.28 and 2.29. Moreover, thanks to the meta data available for the facilities layer, the application could also apply its own security check.

As depicted in Figure 2.28 the V2X message payload is created on application or facilities layer. On network & transport layer a basic network header is created as well as a common

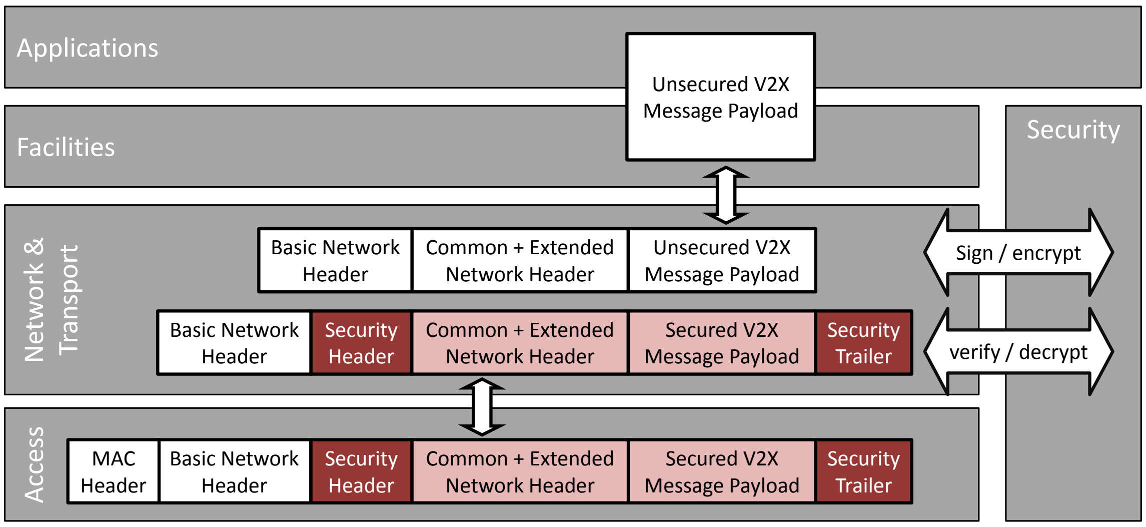


Figure 2.28: Security processing in the communication stack

and optionally an extended network header. The basic network header contains uncritical mutable information that changes while messages are forwarded over multiple hops [15]. As a result, this header is not protected by the security. The common and extended network header, however, contain critical information that is protected by the security. The VSS generates the security header and trailer and integrates them into the V2X packet according to ETSI EN 302 636-4-1 [15]. Finally, the packet is extended by the MAC header on access layer.

Figure 2.29 illustrates the content of a V2X packet in more detail. In general the critical content like identifiers and mobility data has to be protected by the security. In addition to the security itself the extended network layer header may contain position data as well as the CAM and DENM.

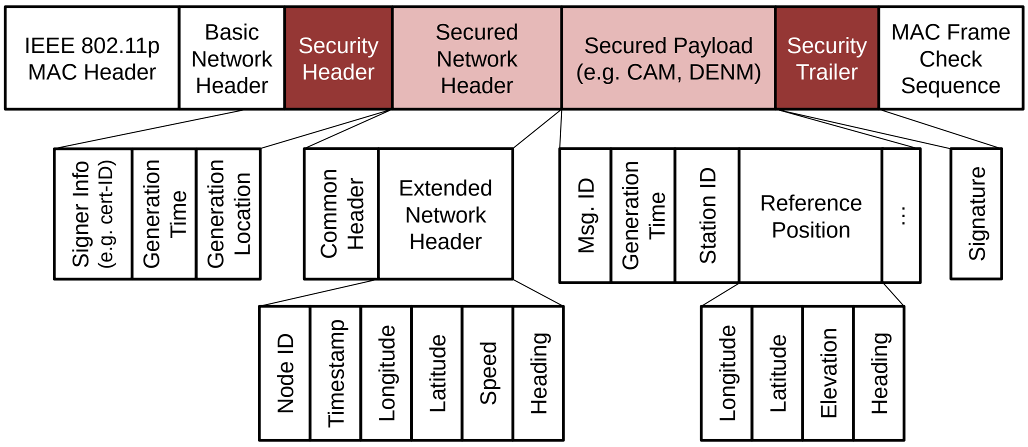


Figure 2.29: Integration of security header into V2X packet structure

#### 2.9.1.3 Convergence Layer

The Convergence Layer (CL) is the connector between the communication system and the VSS. It is used in the PRESERVE VSA in order to allow a flexible integration into different V2X communication stacks and thus allowing an easier reuse of the VSS. The basic concept has been derived from the SeVeCom project.

The Convergence Layer consists of two APIs: an external API to the communication system and an internal API connected to different PRESERVE modules.

As shown in the sequence diagram of outgoing and incoming messages in Sections 2.8.1.1 and 2.8.1.2 respectively, the CL is on the critical path. Therefore, low latency requirements are given. Additionally, high access frequency are defined in the requirements analysis of the PRESERVE deliverable D1.1 [36, Section 3.1.8].

In the following, the internal connection points with other VSS modules are listed as well as example data flow descriptions. Furthermore, internal dependencies to connected modules are described as also illustrated in Figure 2.30. The external and internal interfaces of the CL are further detailed in the remaining of this section.

We note that the CL provides both the stack- and the VSS-controlled API at the external interface and system developers integrating the PRESERVE VSS are free to choose either approach.

#### Connection to other Modules of the PRESERVE Architecture

- External components such as applications and the communication stack
- Secure Communication Module (SCM)
- Security Event Processor (SEP)

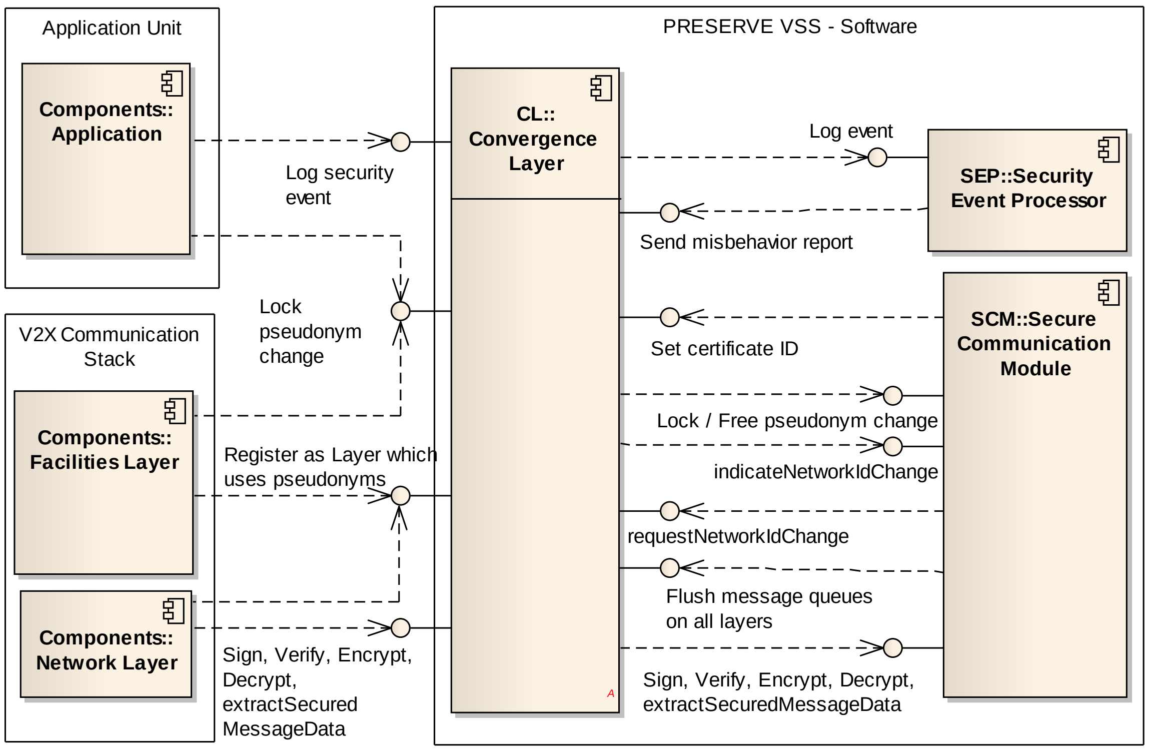

Application Unit
Components:: Application
V2X Communication Stack
Components:: Facilities Layer
Components:: Network Layer
PRESERVE VSS - Software
CL:: Convergence Layer
SEP::Security Event Processor
SCM::Secure Communication Module
Log security event
Lock pseudonym change
Register as Layer which uses pseudonyms
Sign, Verify, Encrypt, Decrypt, extractSecured MessageData
Log event
Send misbehavior report
Set certificate ID
Lock / Free pseudonym change
indicateNetworkIdChange
requestNetworkIdChange
Flush message queues on all layers
Sign, Verify, Encrypt, Decrypt, extractSecuredMessageData

Figure 2.30: Integration of the Convergence Layer into the on-board security architecture

## Data Flow

- Applications or the facilities layer are able to lock the pseudonym change in critical situations. The CL provides an external interface and forwards the information about locks to the PMM.
- The interface for message protection is provided as API and is described in more details in this section. Related process descriptions for using this API for message processing can be found in Sections 2.8.1.1 and 2.8.1.2.
- The CL provides an external interface where the different layers can register call-back functions that are called in case of a pseudonym change. The appropriate internal interface is used indirectly by the Pseudonym Management Module (PMM) in order to execute a pseudonym change. As PMM has no direct connection to the CL the SCM is used as proxy.
- The CL provides on the external side an interface for reporting application specific misbehavior detection events. These events are forwarded to the SEP module.
- The CL provides an internal interface for using the communication stack by VSS components. This can be done in form of a normal application. The access to the communication layer is used by the PMM in order to request new pseudonym certificates from the PKI and by SEP in order to send misbehavior reports to infrastructure services.

**Dependencies** The CL strongly depends on:

- SCM in order to sign / verify, encrypt / decrypt messages
- SCM in order to forward the pseudonym change lock request from application or facilities layer.
- SEP in order to forward security reports from applications

**External Interface Summary** The PRESERVE external API consists of two groups of functions: Packet processing and auxiliary functions. We consider the network layer to be the appropriate layer to do the security processing of incoming and outgoing packets in V2X communication systems as motivated in Section 2.9.1. We envision the security subsystem to be a modular addition to the normal packet processing pipeline of V2X communication systems. The API should be reasonably small and the provided service should be self contained.

**Packet Processing** It is necessary to provide different modes of operation in order to satisfy the different needs of stack implementers. While it would be assumed that most implementers should prefer an isolated modularized approach it is recognized that stack implementers might want to follow a more imperative style. In the latter style the control over the processing of packets on the critical path of execution is conceptually not relegated to an external component. Processing packets can be achieved in the PRESERVE VSA in two ways:

- **VSS Controlled Packet Processing:** In this usage pattern the network layer simply has to hand-over incoming and outgoing packets to the VSS to do whatever is necessary to ensure the security and privacy of V2X communication. The details are completely abstracted and isolated from other subsystems. We call this the *VSS controlled* scheme.
- **Stack Controlled Packet Processing:** The other usage pattern allows the network layer to manually drive the processing of packets. This would involve encryption and signing of outgoing packets and verification and decryption of incoming packets. Decisions about what is necessary to do with packets and how to react to unexpected events have to be made in the network layer. We refer to this mode of operations as *stack controlled* processing.

**Auxiliary Functions** A number of auxiliary functions enable various interactions between the host (communication stack and V2X applications) and the VSS. These interactions include locking pseudonym changes and subscribing to security event notifications. In addition to providing services to the host, the VSS also needs to interact with services of the host. In particular the VSS needs the ability to send and receive packets, e.g. pseudonym refill requests or certificate revocation list updates. The VSS also need the ability to freeze or flush other subsystems while pseudonyms are changed.

**External Interface Description** The following API description is divided in two sections: The main packet processing functions and a number of auxiliary functions. For the packet processing functions the VSS controlled processing is offered using an securityHookOutgoing / securityHookIncoming approach and a manual mode exposing individual functions of the packet processing steps. The following Tables 2.9, 2.10, and 2.11 give an overview of available external API functions. The columns *pattern* and *type* declare the direction of the API call. The abbreviations used here are based on the ETSI reference architecture [5] shown in Figure 2.25 on page 65 and ETSI Service Access Point (SAP) definitions [10].

As introduced in Section 2.9.1.3 the security packet processing can be done following two different approaches. Interfaces of the VSS controlled packet processing approach are shown in Table 2.9. The registration of the VSS controlled packet processing by using hooks can be done in different ways that are out of scope of this document.

Table 2.9: Methods of VSS controlled packet processing approach of external CL API

| Function name | Pattern | Type | Description |
|---|---|---|---|
| **Networking Layer (SN)** | | | |
| securityHookOutgoing | SN->VSS | REQUEST | Process an outgoing packet |
| securityHookIncoming | SN->VSS | REQUEST | Process an incoming packet |

Interfaces of the stack controlled packet processing approach are shown in Table 2.10.

Table 2.10: Methods of stack controlled packet processing approach of external CL API

| Function name | Pattern | Type | Description |
|---|---|---|---|
| **Networking Layer (SN)** | | | |
| sign | SN->VSS | REQUEST | Sign a payload |
| verify | SN->VSS | REQUEST | Verify a payload |
| encrypt | SN->VSS | REQUEST | Encrypt a payload |
| decrypt | SN->VSS | REQUEST | Decrypt a payload |
| extractSecuredMessage-Data | SN->VSS | REQUEST | Extract a payload from a secure message format |

The auxiliary functions of the Convergence Layer are shown in Table 2.11.

Table 2.11: Methods of auxiliary functions of external CL API

| Function name | Pattern | Type | Description |
|---|---|---|---|
| **Facilities Layer (SF)** | | | |
| lockPseudonymChange | SF->VSS | REQUEST | Lock pseudonym changes for a given number of milliseconds |
| unlockPseudonymChange | SF->VSS | REQUEST | Reset lock of pseudonym changes |

| Function name | Pattern | Type | Description |
|---|---|---|---|
| requestPseudonym-Change | SF->VSS | REQUEST | Initiate a change of pseudonym. Active pseudonym change locks should not be ignored. |
| lockOutgoingMessage-Processing | VSS->SF | COMMAND | Used during the process of changing pseudonym |
| unlockOutgoingMessage-Processing | VSS->SF | COMMAND | Used during the process of changing pseudonym |
| send | VSS->SF | COMMAND | E.g. send pseudonym refill request, CRLs update request |
| receive | SF->VSS | REQUEST | E.g. receive pseudonym refill response, CRLs |
| **All Layers (S*: SI, SN, SF)** | | | |
| changePseudonym | VSS->S* | COMMAND | Change pseudonym |
| flushOrDropMessage-Queue | VSS->S* | COMMAND | Used during the process of changing pseudonym |
| registerNotification-PseudonymChange | S*->VSS | REQUEST | Allow clients/layers to register additional callbacks to process pseudonym change events |
| logSecurityEvent | S*->VSS | REQUEST | Send a notification about a security event to the VSS |

## 2.9.2 Meta Data

From an architectural perspective the different layers of the communication stack (cf. Fig. 2.25) are independent from the data of other layers. In an optimal security solution each layer has to cryptographically protect its own data by adding a dedicated security header. In practice this strategy would enlarge the packet size dramatically and prevent a reliable high frequent broadcast communication. Consequently, only a single security header is considered per packet as depicted in Fig. 2.29 on page 71. In this architecture, the process of securing outgoing messages and verifying incoming messages is done on the network layer, as discussed in Section 2.9.1. Consequently, it is necessary that information can be exchanged between the layers regarding the requested security operations and their results. The following example expresses this necessity.

**Example 2.1** *An emergency vehicle is permitted to send special messages indicating for example activated blue lights. But most of the times the vehicle is sending normal V2X messages that do not contain information about the sender's special permissions. Therefore, the vehicle is equipped with two types of pseudonym certificates that contain different permissions. If the vehicle is operated in civil mode then the pseudonym has to be changed as soon as the blue light is activated. In this case an application on application layer creates a V2X message that has to be signed by a certificate with special permission on security layer. In order to inform the security that is applied on network layer about this* *requirement, the application appends meta data to the message and sends it downwards the communication stack. The VSS on network layer receives the message with its meta data and can react appropriately by signing with a valid certificate that contains all necessary permissions. The VSS of receiving ITS stations verify the message on network layer and create meta data that contain the permissions from the sender's certificate as well as information regarding the verification process. Subsequently, the verified message and the meta data is forwarded upwards the communication stack to the application layer. The application is then able to check whether the sender was permitted to send this special kind of message.*

Motivated by this example it is obvious that additional operation information has to be exchanged between the layers of the communication stack as shown in Figure 2.31. In this VSA two ways are described to exchange meta data.

- Collection of station-related meta information in the management layer
- Bundling message-related meta data with the V2X message that is transmitted downwards and upwards the communication stack

The station-related meta information is static. In general the identifiers of the neighboring stations can be considered as static over a specific period of time. Consequently, the content of the pseudonym certificate of the sender can be assumed to be static for this time. According to the Standards Profile of the Car-to-Car Communication Consortium [39] the identifiers (StationId in CAM/DENM, GeoNetwork source address) shall be derived from the pseudonym certificate ID. Identifiers having a size equal to 64 bit shall use the pseudonym certificate ID (of type HashedId8 according to ETSI TS 103 097 [14]) directly (such as GeoNetwork source address). For shorter identifiers, the least significant bytes of the pseudonym certificate ID shall be used as identifier (such as the StationId in CAM/DENM). As a result, the receivers are able to link the GeoNetwork source address and the StationId in CAM or DENM to the certificate ID. This ID can be used to manage the station-related meta information in the management layer. As soon as a V2X message is received the network & transport layer triggers the VSS to verify the signature. The security subsystem extracts all relevant information from the security header and the certificate and hands it over to the management layer via the MS interface. The management layer collects all information related to a specific node by using the certificate ID as index. After verification of the packet the network & transport layer extracts the station-related information from the network headers and hands it over to the management layer via the MN interface using the GeoNetwork source address as identifier. Since the GeoNetwork source address is derived from the certificate ID the management layer is able to find the correct index of the station. In the same way the facilities layer extracts the station-related information from the CAM or DENM and hands it over to the management layer via the MF interface using the StationId as identifier. As shown in Figure 2.31 the access layer is not able to hand over station-related information to the management layer. This is because the MAC address is not linked to the certificate ID on sender side and therefore the access layer on receiver side cannot identify the correct index. Moreover, the data of the access layer is not protected by the security. Consequently, the receiver cannot trust this information.

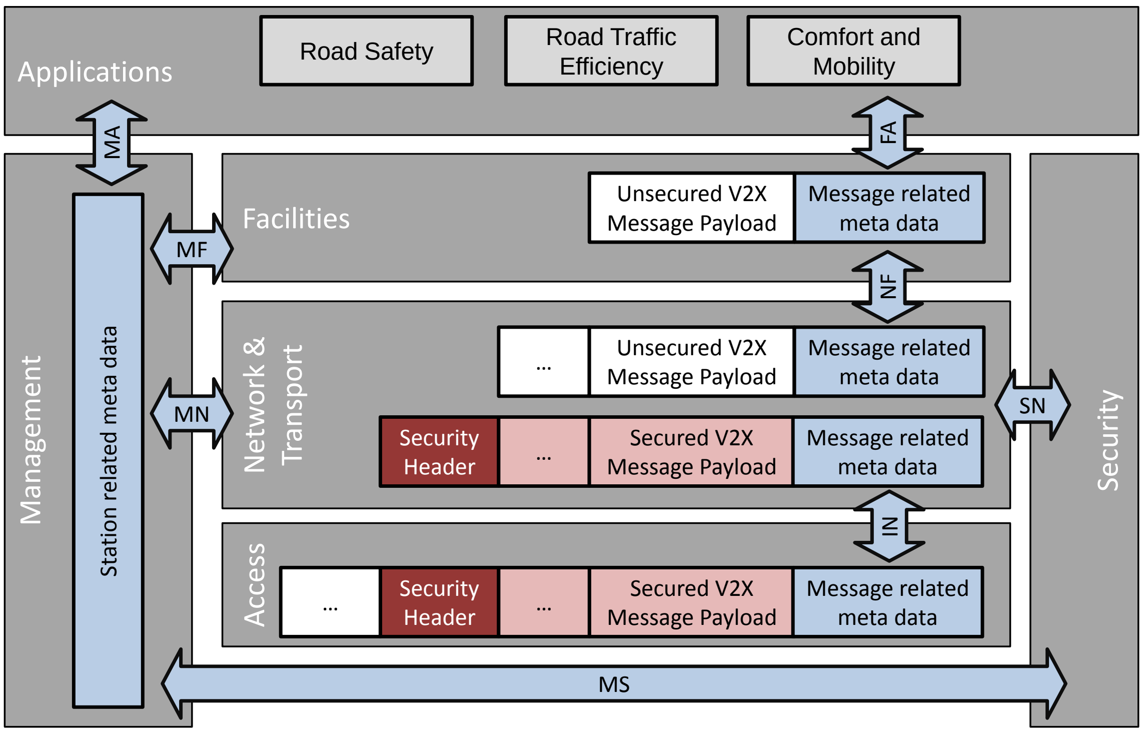


Figure 2.31: Meta data exchange between security layer and application layer

In addition to the collection of station-related meta data in the management layer message-related meta data is bundled with the V2X messages and handed over through the communication stack via interfaces IN, SN, NF, FA. For this bundling of meta data to the message no identifier is needed. In case of message reception relevant meta data from the access layer is bundled with the message while it is handed over to the network & transport layer via the IN interface. Subsequently the security provides the message-related meta data. The network & transport bundles the meta data of the security and the meta data extracted from the network headers to the message that is handed over to the facilities layer via the NF interface. Finally, the facilities layer provides the message and the message-related meta data to the applications via the FA interface. In this concept only the provider and consumer of meta data must be able to interpret this information. The content of meta data is flexible with respect to the number of elements and their extensibility with new elements. In general, meta data can be structured as a triple *name, type, value* that can be extended also in runtime if the type of value is defined previously.

#### 2.9.2.1 Meta Data Elements

Table 2.12 and 2.13 presents possible meta data elements that are transmitted upwards the communication stack from the access layer to the application layer. Additionally, meta data elements are created by a applications and transmitted afterwards from application layer down to the network layer in order to use them in the VSS.

Table 2.12: Station-related meta data elements transmitted from security (network layer) to upper layers (e.g. georouting or application layer)

| | Name | Type | Description |
|---|---|---|---|
| **Station-related** | aid | Array of AIDs extended by Service Specific Permissions (SSPs) | List of application IDs and service specific permissions extracted from the certificate. The application at the receiver can check whether the sender has appropriate permissions to send specific messages. |
| | sender-Reputation | Double | A plausibility check inside the VSS can provide reputation information to the facilities or application layer regarding trustworthiness of neighboring ITS station. |
| | localNodeID | Long | A local node ID could be provided by a message plausibility check that tracks all neighboring vehicles and is able to detect a pseudonym change in the single-hop communication range. This local node ID is only available for local applications and must not be provided to external entities. The plausibility check could be applied on network layer or application / facilities layer. |

Table 2.13: Message-related meta data elements transmitted from security (network layer) to upper layers (e.g. georouting or application layer)

| | Name | Type | Description |
|---|---|---|---|
| **Message-related** | security-Processing-Result | Flags | The application at the receiver can check whether incoming messages are verified correctly or message was decrypted by VSS. It is maybe important for an application to known whether the received message was encrypted or not. All necessary flags have to be defined later in more detail. The IEEE 1609.2 format for flags defines only the representation for transmission as byte stream. |
| | genera-tionTime | Time64With-Confidence | A plausibility check on application or facilities layer can compare the generation time of message payload (CAM, DENM) with time from security header. |
| | expiryTime | Time64 | A plausibility check on application or facilities layer can compare the expiry time of message payload (CAM, DENM) with time from security header. |

| | Name | Type | Description |
|---|---|---|---|
| Message-related | location | ThreeDLocation-AndConfidence | A plausibility check on application or facilities layer can compare the location of message payload (CAM, DENM) with location from security header. |
| | message-Plausibility | Double | A plausibility check inside the VSS can provide plausibility information to the facilities or application layer regarding trustworthiness of a single incoming message. |
| | privacy-Processing-Result | Flags | The application at the receiver can check whether incoming messages are following privacy policies. All necessary flags have to be defined later in more detail. |

Table 2.14: Message-related meta data elements transmitted from upper layers (e.g. application layer or facilities layer) to security (network layer)

| | Name | Type | Description |
|---|---|---|---|
| Message-related | aid | Array of AIDs extended by Service Specific Permissions (SSPs) | List of required permissions. For example, an application on facilities layer may decide whether emergency lights are active which could lead to a pseudonym change in the VSS. |
| | targetStationId | Long | If a messages should be encrypted the address of the target has to be transmitted to the VSS. As only the station ID of the target is known by the sender application this value should be used to get the appropriate certificate for encrypting the message inside the VSS. |
| | securityPro-cessingNeeds | Flags | The application at the sender can provide needs for security message processing (e.g. add generationTime, expiryTime or location to security header). All necessary flags have to be defined by implementers in more detail. |
| | privacyPro-cessingNeeds | undefined | May contain application specific certificates and privacy related data such as policies. |

#### 2.9.2.2 Concept for Stack Integration

The security meta data could be transmitted as data blob (block of bytes), attached to messages, passing the different layers of the communication stack. There is no need for the communication stack to access or use the attached security meta data - hence the internal data structure can be opaque to the communication stack. The data blob can simply be handed over by the layers inside the stack. Only the producer and the consumer of meta data (e.g. VSS and applications) must be able to understand the format. The format itself has to be flexible and extensible, as new applications may need to transfer different meta data from and to the VSS. Another requirement is that it is an efficient format regarding CPU load for generating and interpretation. An exemplary generic structure is shown in listing 2.1.

While such a “security-only” meta-data facility can be implemented, we want to stress that we consider the concept of meta-data for packets and stations as a generic capability that can be useful in many ways, not only limited to security. Therefore, we argue that the communication stack itself should provide this as a service that the VSS and other components can use.

```
enum {
        mdet_Flags(0),
        mdet_AidSspArray(1),
        mdet_Time64WithConfidence(4),
        mdet_Time64(5),
        mdet_ThreeDLocationAndConfidence(6),
        mdet_Integer(7),
        mdet_Double(8),
        mdet_Long(9),
        (2^8-1)
} MetaDataElementType;

struct {
        opaque name<var>;
        MetaDataElementType type;
        select (type) {
                ...
        }
} MetaDataElement

struct {
        MetaDataElement  <2^8-1>;
} MetaData
```

Listing 2.1: Structure for exemplary meta data

#### 2.9.2.3 Upwards from Security (Network Layer) to Facilities or Application Layer

After reception of a message, the communication stack (network layer) calls the convergence layer API with verify.request() as displayed in process for message receiving. After verifying a message the convergence layer creates the station and message-related meta data and provides the message-related data as byte array via method verify.confirm() to the network layer. The station-related information might be provided to the management layer implementation. The message-related meta data in the contrary is attached to the message and transmitted with it upwards to the applications. The applications are then able to parse the security meta data on their own or the facilities layer provides an API that reads and provides the meta data. In addition, the applications are able to request station-related information from the management-application interface (MA) as depicted in Figure 2.31.

#### 2.9.2.4 Downwards from Facilities or Application Layer to Security (Network Layer)

The meta data transmission from an application or the facilities layer can be realized in a similar way. The application creates the meta data byte array and provides it to the communication stack. It is necessary that the interface on application / facilities layer provides a possibility for adding meta data. The message with attached meta data then travels the stack downwards. On network layer the stack calls the convergence layer API with sign.request() as displayed in process for message sending and Figure 2.31. The convergence layer parses the meta data and uses the internal API to call the appropriate methods inside the VSS.

### 2.9.3 Stack Parallelism

In V2X communication, the state-of-the-art digital signature scheme ECDSA is used to ensure the authenticity of each message. The point multiplication as one of the main operations in ECDSA is a very time consuming task. Not only in software based solutions but also in the much faster hardware-based solutions, this operation is the bottleneck of the overall message processing because the signature of every incoming message must be verified. As stated in the PRESERVE deliverable D1.1 [36], a performance of about 1000 verifications per second is required to handle the received V2X messages even in heavy traffic situations. Hardware implementations with highly optimized ECC accelerators running on a high performance FPGA are able to do 400 verifications per second, c.f. the EVITA FPGA. Besides the fact that 400 verifications per second are not sufficient, high performance FPGAs can't be used in large-scaled field tests because they are way too expensive.

One main target of PRESERVE is it to develop a cheap and scalable close-to-market ASIC that can be used for field tests. Providing 1000 verifications per second under low cost conditions is a really challenging task. Improving the performance to this level can be done in two ways. First, one can increase the clock rate, which is limited by the technology and node size. However, using smaller node sizes increases costs and has physical limitations. This is the reason why this approach is neither cost efficient nor feasible and in particular not in line with the PRESERVE targets.

The second approach is to parallelize the critical operations, i.e. the ECC point multiplications. This approach is also state-of-the-art and common sense in many other areas of today's computer technology. For example, a standard desktop PC today will be using a CPU with 4 cores running at 3.5 GHz rather than one core at 14 GHz to provide the performance needed, work power efficient and be cheap. This fact can be transferred one-to-one and reflects the problem of the message verification in a V2X HSM. The parallelized approach is essential to reach the performance requirements efficiently with respect to cost, power consumption, size etc.

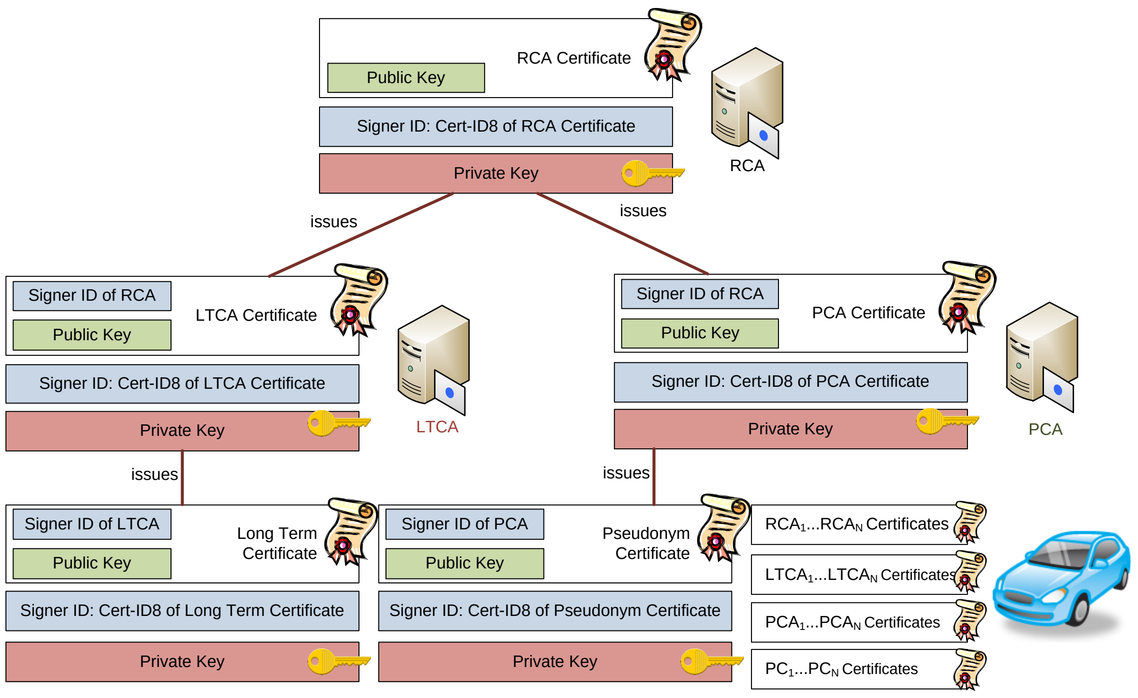
This conclusion is in-line with the results of the PRESERVE / C2C-CC Security Architecture Workshop from June 2013 and is general consensus also among industry experts.

The multi-core approach implicitly creates a requirement for the communication stack to fully exploit the performance capabilities of the HSM. The communicate stack must implement message pipelining, i.e. asynchronously trigger a certain number of cryptographic operations like signature verifications corresponding to the number of crypto cores in the HSM. We stress that this is different from a random parallel execution and should be comparatively simple to implement. As pipelining should be considered a standard technique in today's ICT, we do not consider this technique as an unjustified or extreme burden to extend a communication stack appropriately.

# 3 Security Infrastructure

As explained in the introduction of Section 1.2 the trust between message sender and receiver is based on digital certificates that are provided by a PKI. In this section the PRESERVE infrastructure is discussed in general. The PKI concept of PRESERVE aims to be compatible with ETSI [7] and uses certificate formats defined by IEEE 1609.2 [22] and ETSI TS 103 097 [14].

The general PKI used in the VSA consists of the three entities RCA, LTCA and PCA. The Root CA is the trust anchor of the PKI and the certificate of the RCA is signed by itself. As shown in Figure 3.1, the RCA certificate consists mainly of a public key plus additional information such as validity and permissions. With a hash function a digest of the certificate is created that is used subsequently as cert-ID. As previously mentioned, the RCA certificate is self-signed, that is why the certificate contains no signer ID. Fitting to the public key, a private key created by the Root CA is used to sign other CAs or other relevant data. The RCA certificate and the cert-ID are public and must be available to all ITS stations in the network.



RCA Certificate
Public Key
Signer ID: Cert-ID8 of RCA Certificate
Private Key
RCA
issues
issues
Signer ID of RCA
LTCA Certificate
Public Key
Signer ID: Cert-ID8 of LTCA Certificate
Private Key
LTCA
Signer ID of RCA
PCA Certificate
Public Key
Signer ID: Cert-ID8 of PCA Certificate
Private Key
PCA
issues
issues
Signer ID of LTCA
Long Term Certificate
Public Key
Signer ID: Cert-ID8 of Long Term Certificate
Private Key
Signer ID of PCA
Pseudonym Certificate
Public Key
Signer ID: Cert-ID8 of Pseudonym Certificate
Private Key
RCA$_1$...RCA$_N$ Certificates
LTCA$_1$...LTCA$_N$ Certificates
PCA$_1$...PCA$_N$ Certificates
PC$_1$...PC$_N$ Certificates

Figure 3.1: Credentials used in the V2X Public Key Infrastructure

With the private key of the RCA, certificates for additional CAs can be issued. The LTCA is responsible for management of long term certificates of the ITS stations that contain identifying information. The PCA is responsible for issuing pseudonym certificates that do not contain any identifying information and are reduced to a minimum of size. In order to issue CA certificates by the RCA, the LTCA and PCA create independently public and private key pairs. The public key is transmitted to the RCA where an appropriate certificate is generated. The permissions of the LTCA and PCA as well as the public key are stored in the unsigned certificate format. Subsequently, the RCA adds its own cert-ID as signer ID and signs the certificate with its private key. The signed certificate is then returned to the respective CA. Equally to the Root CA, the LTCA and the PCA create a cert-ID out of their own certificate and publish the certificate afterwards. The private key must be protected particularly in order to avoid misuse of the PKI.

In order to request pseudonym certificates from the PCA, the ITS station has to be equipped with a valid long-term certificate. This is issued by the LTCA. Compared to the CA certificates, the LTC contains the signer ID, in this case the LTCA cert-ID, and further information such as validity and permissions. If the ITS station has received an LTC it is able to request several short-term certificates from the PCA that can be used in V2X communication. A PC contains the cert-ID of the PCA as signer ID, the public key and further certificate specific data such as permissions. Additionally to the LTC and the PCs every ITS station has to store the certificates of the CAs (i.e. from the RCA, LTCA and PCA). While communicating with other stations via ad-hoc network (i.e. ITS-G5A, ITS-G5B or ITS-G5C) the certificates of all message originators have to be stored temporarily.

## 3.1 Interfaces between ITS Station and Infrastructure

Between the ITS station and the PKI, an interface is used to exchange security data. The following processes have to be considered between the ITS station and the LTCA:

- Registration of VSS in order to initialize the security subsystem for further interactions with the PKI afterwards
- Long-term certificate request, response, error, acknowledgment

Between the ITS station and the PCA the following processes must be foreseen:

- Pseudonym certificate request, response, error, acknowledgment
- CRL request, response

As shown in Figure 3.2, the ITS station sends a request to the Pseudonym CA. This request includes the encrypted signer ID of its LTC, one or several public keys, permissions or restrictions and an ID or address of the Long-Term CA. Due to encryption, the PCA is not able to create a link between the pseudonym and the long-term-ID of the requester. Also the Long-Term CA is not able to create such a link because the Pseudonym CA does not forward the public key or pseudonym certificate. This split of power between PCA and LTCA is necessary due to privacy protection requirements. Nevertheless, a trusted network connection between both CAs is necessary.

If the VSA is applied in an FOT, the signer ID can alternatively be transmitted unencrypted so that the Pseudonym CA is able to operate a database by storing links between the long-term certificate and requested pseudonym certificates. This can be used for debugging and testing purposes.

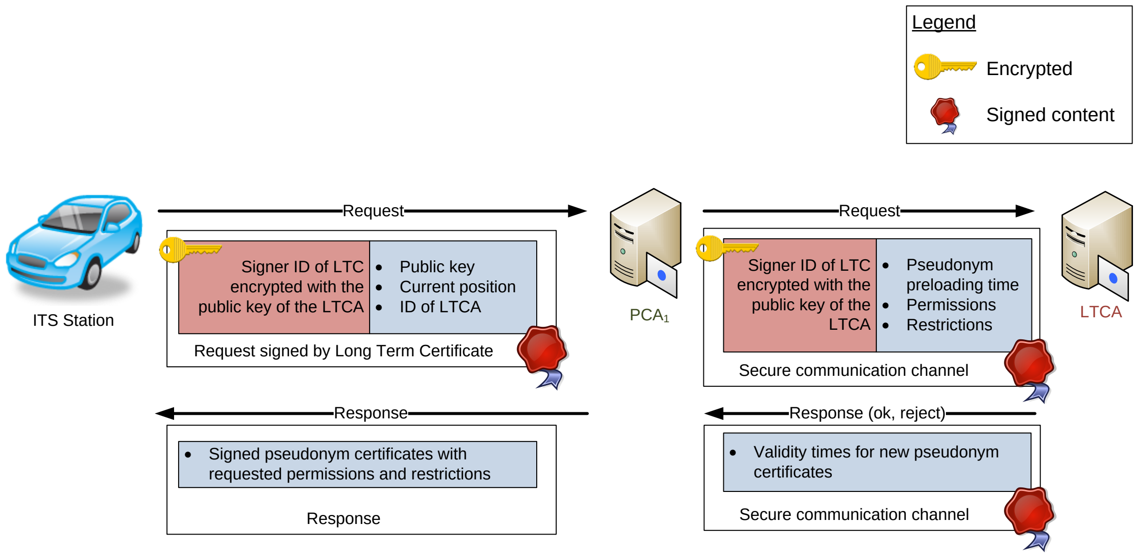

Legend
Encrypted
Signed content
ITS Station
Request
Signer ID of LTC encrypted with the public key of the LTCA
Public key
Current position
ID of LTCA
Request signed by Long Term Certificate
PCA1
Request
Signer ID of LTC encrypted with the public key of the LTCA
Pseudonym preloading time
Permissions
Restrictions
Secure communication channel
LTCA
Response
Signed pseudonym certificates with requested permissions and restrictions
Response
Response (ok, reject)
Validity times for new pseudonym certificates
Secure communication channel

Figure 3.2: Request of Pseudonym Certificates

The Pseudonym CA sends a request with the (encrypted) signer ID of the requester, a preloading time and the region ID to the Long-Term CA for verification. The LTCA maintains a database that stores the timestamp until a vehicle has valid pseudonyms. Only if a valid LTC can be found and for the requested time not the maximum number of permitted pseudonyms have been acknowledged previously, the Pseudonym CA gets a positive response from the Long-Term CA. The described process is based on constantly available PCA and LTCA. An offline generation of pseudonyms on the PCA without contact to the responsible LTCA is not possible in this concept as the permission and the timestamps for pseudonym certificate validity have to be requested individually for every pseudonym public key from the LTCA.

## 3.2 Revocation

A fast and efficient revocation of pseudonym certificates in the VSA is difficult due to communication restrictions between vehicles and Pseudonym CA and the possibly large number of revoked pseudonym certificates. As a result, the PKI does not consider CRL distribution within the vehicular network for pseudonym certificate revocation. A revocation of ITS stations can only be done by rejecting the request for new pseudonym certificates.

In the PRESERVE PKI the Long-Term CA links the revocation information of the ITS station to its LTC. If a station requests new pseudonym certificates then the Pseudonym CA forwards the request to the respective Long-Term CA which checks the authorization of the requester. The Long-Term CA is operating a database that contains information about the validity of the long-term certificate and currently allowed pseudonym requests linked with the PCA. A revoked long-term certificate can be marked as active or inactive in this database. In case of revocation the Long-Term CA provides information about revocation to the Pseudonym CA whereupon the Pseudonym CA reject the PC update request.

For revocation of Pseudonym CA, Long-Term CA and Root CA, the distribution of a CRL is proposed because CA certificates are predominantly valid for a long period of time (i.e. several years). All revoked CAs are added to a CRL that has to be distributed to all ITS stations in the V2X communication network. The ITS station can request the latest CRL from the PCA at every pseudonym refill process. As the revocation of CA certificates should only be used in extreme cases (e.g. successful attack on the PKI) the CRL contains probably very few entries and changes very seldom. Nevertheless, a CA certificate that is compromised or not trustworthy due to other reasons is manually revoked by the PKI administration. Afterwards the certificate ID of the affected CA is added to the CRL and signed by the Root CA. With incremental serial numbers the receivers (LTCA, PCA and ITS stations) are able to check if the locally stored CRL is up to date or must be exchanged. Details regarding the CRL format can be found in IEEE 1609.2 v2 [22, Section 5.3.39].

## 3.3 Pseudonym Resolution

As described in Section 2.4.4 privacy protection is an important issue in V2X communication. Also in the backend privacy should be preserved. Therefore, no entity in the backend (i.e. in the PKI) should be able to link a pseudonym to its owner. As described in the proposed PKI concept of the Car 2 Car Communication Consortium [2], the PCA should only be able to see information related to the pseudonym and the LTCA should only be able to see information related to the requester but no information related to a specific pseudonym. With this split of powers, both entities are not able to create a link between the pseudonym and its requester. If non-repudiation requirements are given (e.g. for law enforcement), a third party could be integrated into the process of pseudonym certificate requests in order to ensure controlled resolution of pseudonyms by considering required privacy protection requirements. Additionally, further cryptographic keys can be added to the pseudonym certificates that are used to enable the linking between long-term certificate and pseudonym in special cases as described in [31].

Nevertheless, in an FOT it may be necessary to have this link available due to test evaluation requirements. As result, according to the field of application the pseudonym certificate request process has to be adapted. If the FOT requires to resolve the pseudonym owner then the VSS has to provide the long-term ID in clear text so that a database can be maintained in the PKI that stores a link between PC and LTC. Otherwise, if there is no requirement by the FOT to know the pseudonym owner for evaluation the VSS should transmit the long-term identifiers in an encrypted way to the PCA as described in Section 2.5.1.3.

# 4 Conclusion

The VSA described in this document aims to combine security mechanisms into a flexible and scalable security architecture that is able to protect V2X communications against relevant attackers. This security architecture is designed to be used in different future close-to-market FOTs. This architecture is based on the threat analysis provided in the PRESERVE deliverable D1.1 [36] which identified most relevant security aspects. Due to the re-usage of results from several previous projects, adequacy and effectiveness of specific security solutions are already proven.

The VSA described in this document integrates security solutions from the following three cornerstones:

- **V2X communication security** considered in this VSA are based primarily on results of the SeVeCom project. The sender authenticity and message integrity are most relevant for secure and trustworthy V2X communication using ITS-G5A as the majority of messages (i.e. CAMs and DENMs) are exchanged in broadcast mode. Nevertheless, these generic security requirements impose several complex mechanisms on the ITS station as well as in the infrastructure. The integration of hardware security mechanisms is necessary for vehicles and RSUs in order to protect private keys and certificates against manipulation or extraction and to accelerate cryptographic operations. In the infrastructure, a PKI must be implemented that provides certificates used to realize sender authenticity between potentially unknown communication endpoints. A generic PKI structure has been specified by the C2C-CC [2] where PRESERVE partners were involved in lead positions. The fundamentals of the certificate management in the PKI and inside the ITS station are based on proposals of the SeVeCom project.
- **On-board security** considered in this VSA is based on EVITA project results. The authenticity and integrity of data contained in CAMs and DENMs bases on security mechanisms that protect the communication with on-board sensors and ECUs. The HSM used for V2X communication security can also be used to protect this on-board communication. Therefore, all on-board components equipped with EVITA hardware can be connected in a secure way to the PRESERVE VSS that is equipped with an EVITA HSM too.
- **V2X communication privacy protection** considered in this VSA is based on PRECIOSA project results. Since vehicle drivers have strong demands to be untraceable in the V2X network the integration of privacy protection mechanisms are essential in the presented VSA. Respective privacy enhancing technologies must be deeply

integrated into the VSS of the ITS station but also into the PKI solution of the infrastructure. Furthermore, strong policy based privacy protection can optionally be integrated using the PeRA privacy policy enforcement framework.

The resulting VSA benefits from respective preparatory work and provides consolidated interfaces to integrate the VSS into ITS stations if they base on the ETSI communication reference architecture [5]. Furthermore, adaptability aspects of interfaces are considered in the VSA in order to be flexible regarding future requirements.

# Bibliography

[1] N. Bißmeyer, C. Stresing, and K. Bayarou, “Intrusion detection in vanets through verification of vehicle movement data,” in *Second IEEE Vehicular Networking Conference*, vol. Second IEEE Vehicular Networking Conference, December 2010.

[2] N. Bißmeyer, J. P. Stotz, H. Stübing, E. Schoch, S. Götz, and B. Lonc, “A generic public key infrastructure for securing car-to-x communication,” in *18th World Congress on Intelligent Transportation Systems*. ITS America, October 2011.

[3] N. Bißmeyer, H. Stübing, M. Mattheß, J. P. Stotz, J. Schütte, M. Gerlach, and F. Friederici, “simTD security architecture: Deployment of a security and privacy architecture in field operational tests,” in *7th Conference: escar - Embedded Security in Cars*. isits International School of IT Security, November 2009.

[4] ETSI - European Telecommunications Standards Institute, “Intelligent transport systems (ITS); vehicular communications; basic set of applications; definitions,” ETSI, Technical Report TR 102 638, June 2009, v1.1.1.

[5] ——, “Intelligent transport systems (ITS); communications architecture,” ETSI, European Norm EN 302 665, September 2010.

[6] ——, “Intelligent transport systems (ITS); european profile standard for the physical and medium access control layer of intelligent transport systems operating in the 5 ghz frequency band,” ETSI, European Standard ES 202 663, January 2010, v1.1.0.

[7] ——, “Intelligent transport systems (ITS); security; security services and architecture,” ETSI, Technical Standard TS 102 731, September 2010.

[8] ——, “Intelligent transport systems (ITS); vehicular communications; basic set of applications; part 2: Specification of cooperative awareness basic service,” ETSI, Technical Standard TS 102 637-2, April 2010.

[9] ——, “Intelligent transport systems (ITS); vehicular communications; basic set of applications; part 3: Specifications of decentralized environmental notification basic service,” ETSI, Technical Standard TS 102 637-3, September 2010.

[10] ——, “Intelligent transport systems (ITS); osi cross-layer topics; part 1: Architecture and addressing schemes,” ETSI, Technical Standard TS 102 723-1, Mai 2011.

[11] ——, “Intelligent transport systems (ITS); security; stage 3 mapping for ieee 1609.2,” ETSI, Technical Standard TS 102 867, 2011.

[12] ——, “Intelligent transport systems (ITS); vehicular communications; geonetworking; part 3: Network architecture,” ETSI, Tech. Rep., 2011.

[13] ——, “Intelligent transport systems (ITS); vehicular communications; geonetworking; part 5: Transport protocols; sub-part 1: Basic transport protocol,” ETSI, Tech. Rep., 2011.

[14] ——, “Intelligent transport systems (ITS); security; security header and certificate formats,” ETSI, Technical Standard TS 103 097, April 2013, v1.1.1.

[15] ——, “Intelligent transport systems (its); vehicular communications; geonetworking; part 4: Geographical addressing and forwarding for point-to-point and point-to-multipoint communications; sub-part 1: Media-independent functionality,” ETSI, European Norm EN 302 636-4-1, June 2013, draft v0.5.1.

[16] I. O. for Standardization, “Open systems interconnection - basic reference model,” International Organization for Standardization, Tech. Rep. ISO/IEC 7498-1, 1994.

[17] M. Gerlach, “Trusted ad hoc communications for intelligent transportation systems,” Ph.D. dissertation, Technische Universität Berlin, 2010.

[18] P. Golle, D. Greene, and J. Staddon, “Detecting and correcting malicious data in vanets,” in *VANET ’04: Proceedings of the 1st ACM international workshop on Vehicular ad hoc networks*. New York, NY, USA: ACM, 2004, pp. 29–37.

[19] O. Henniger, L. Apvrille, A. Fuchs, Y. Roudier, A. Ruddle, and B. Weyl, “Security requirements for automotive on-board networks,” in *Intelligent Transport Systems Telecommunications,(ITST), 2009*, 2009.

[20] J. Holle, A. Groll, A. Crespo, H. Cankaya, T. Enderle, N. M. Guire, and A. Platschek, “Oversee deliverable d2.2: Specification of security services incl. virtualization and firewall mechanisms,” Open Vehicular Secure Platform, OVERSEE, Deliverable D 2.2, February 2011.

[21] Y.-C. Hu, A. Perrig, and D. B. Johnson, “Wormhole attacks in wireless networks,” *IEEE Journal on Selected Areas in Communications*, vol. 24, no. 2, pp. 370–380, February 2006.

[22] IEEE, “Draft standard for wireless access in vehicular environments - security services for applications and management messages,” Institute of Electrical and Electronics Engineers, Tech. Rep. 1609.2 - 20011 (D9), May 2011.

[23] IEEE Computer Society, “IEEE standard for information technology – telecommunications and information exchange between systems – local and metropolitan area networks – specific requirements – Part II: Wireless LAN medium access control (MAC) and physical layer (PHY) specifications,” IEEE Std 802.11p, Tech. Rep., 2010.

[24] F. Kargl, S. Dietzel, L. Dölle, J. C. Freytag, M. Kost, A. Kung, Z. Ma, and F. Schaub, “PRECIOSA D7 V2X Privacy Verifiable Architecture,” Tech. Rep., November 2009.

[25] A. Kung, “Deliverable 2.1 security architecture and mechanisms for v2v/v2i,” SeVeCom, Tech. Rep. Projectnumber: IST-027795, February 2008.

[26] T. Leinmüller, E. Schoch, and F. Kargl, “Position verficiation approaches for vehicular ad hoc networks,” *Wireless Communications, IEEE*, vol. 13, no. 5, pp. 16 –21, october 2006.

[27] C. Miller, D. Blazakis, D. D. Zovi, S. Esser, V. Iozzo, and R.-P. Weinmann, *IOS Hacker’s Handbook*. Wiley, 2012.

[28] H. Oguma, A. Yoshioka, M. Nishikaw, R. Shigetomi, A. Otsuka, and H. Imai, “New attestation based security architecture for in-vehicle communication,” in *IEEE Global Telecommunications Conference*. IEEE, December 2008, pp. 1–6.

[29] J. Petit and Z. Mammeri, “Authentication and consensus overhead in vehicular ad hoc networks,” *Telecommunication Systems*, pp. 1–14, 2011, 10.1007/s11235-011-9589-y. [Online]. Available: http://dx.doi.org/10.1007/s11235-011-9589-y

[30] Research and I. T. Administration, “The national its architecture 7.0,” U.S. Department of Transportation, Research and Innovative Technology Administration (RITA), Tech. Rep. 7.0, January 2012. [Online]. Available: http://iteris.com/itsarch/html/entity/paents.htm

[31] F. Schaub, F. Kargl, Z. Ma, and M. Weber, “V-tokens for conditional pseudonymity in vanets,” in *IEEE Wireless Communications and Networking Conference (WCNS)*, 2010.

[32] R. K. Schmidt, T. Leinmueller, E. Schoch, A. Held, and G. Schaefer, “Vehicle behavior analysis to enhance security in VANETs,” in *Proceedings of the 4th IEEE Vehicle-to-Vehicle Communications Workshop (V2VCOM2008)*, 2008.

[33] E. Schoch, “Secure communication in inter-vehicle networks,” Ph.D. dissertation, Ulm university, 2009.

[34] E. Schoch, N. Bißmeyer, H. Stübing, B. Lonc, and S. Götz, “Public key infrastructure - memo,” Car 2 Car Communication Consortium, Tech. Rep., 2011.

[35] G. Steel, “Towards a formal security analysis of the Sevecom API,” in *7th Conference: escar - Embedded Security in Cars*. isits International School of IT Security, November 2009.

[36] J. P. Stotz, N. Bißmeyer, F. Kargl, S. Dietzel, P. Papadimitratos, and C. Schleiffer, “PRESERVE D1.1 Security Requirements of Vehicle Security Architecture,” PRESERVE consortium, Deliverable, July 2011.

[37] C. Tarnovsky, “Hacking the smartcard chip,” in *Blackhat Decipher Security 2010*, February 2010. [Online]. Available: http://www.blackhat.com/html/bh-dc-10/bh-dc-10-briefings.html#Tarnovsky

[38] C. Weiß, “Sichere intelligente mobilität testfeld deutschland; deliverable d21.5; spezifikation der it-sicherheitslösung,” Fraunhofer-Institut SIT, Tech. Rep., 2009.

[39] C. Wewetzer, T. Biehle, A. Festag, T. Leinmueller, T. Buburuzan, N. Sofra, E. Schoch, B. Jungk, L. LIN, K. Sjöberg, and A. Brakemaier, “C2CCC basic system standards profile,” CAR 2 CAR Communication Consortium, Draft 0.5, July 2013.

[40] B. Weyl, M. Wolf, F. Zweers, T. Gendrullis, M. S. Idrees, Y. Roudier, H. Schweppe, H. Platzdasch, R. E. Khayari, O. Henniger, D. Scheuermann, A. Fuchs, L. Apvrille, G. Pedroza, H. Seudié, J. Shokrollahi, and A. Keil, "EVITA Deliverable D3.2: Secure On-board Architecture Specification," EVITA Consortium, Tech. Rep., August 2011.

[41] B. Wiedersheim, Z. Ma, F. Kargl, and P. Papadimitratos, "Privacy in inter-vehicular networks: Why simple pseudonym change is not enough," in *Wireless On-demand Network Systems and Services (WONS), 2010 Seventh International Conference on*, feb. 2010, pp. 176 –183.

[42] G. Yan, X. Chen, and S. Olariu, "Providing VANET position integrity through filtering," *Intelligent Transportation Systems, 2009. ITSC '09. 12th International IEEE Conference on Intelligent Transportation Systems Communication*, pp. 1 –6, October 2009.

[43] G. Yan, S. Olariu, and M. C. Weigle, "Providing VANET security through active position detection," *Computer Communications*, vol. 31, no. 12, pp. 2883 – 2897, 2008, mobility Protocols for ITS/VANET.