CONNECT
CCAM TRUST & RESILIENCE

D3.2

CONNECT Trust & Risk Assessment and CAD Twinning Framework (Initial Version)

Project number:	101069688
Project acronym:	**CONNECT**
Project title:	Continuous and Efficient Cooperative Trust Management for Resilient CCAM
Project Start Date:	1st September, 2022
Duration:	36 months
Programme:	HORIZON-CL5-2021-D6-01-04
Deliverable Type:	Other
Reference Number:	D6-01-04 / D3.2 / 1.0.1 - Revised on April 04, 2024
Workpackage:	WP 3
Due Date:	29th February, 2024
Actual Submission Date:	11th March, 2024
Responsible Organisation:	Ulm University
Editor:	Artur Hermann
Dissemination Level:	PU
Revision:	1.0.1 - Revised on April 04, 2024
Abstract:	This deliverable builds upon D3.1 and describes major building blocks for the standalone trust assessment framework (TAF), the federated TAF and the digital twin version TAF-DT. For the standalone TAF, a fine-grained architecture underlying our current prototype is provided, federated TAF and TAF-DT are described as high-level architectures.
Keywords:	Trust Assessment Framework, Risk Assessment Framework, Digital Twin

Funded by the European Union

Funded by the European Union. Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union or CINEA. Neither the European Union nor the granting authority can be held responsible for them.

Editor

Artur Hermann (UULM)

Contributors (ordered according to beneficiary numbers)

Nikos Fotos, Anna Angelogianni, Thanassis Giannetsos (UBITECH)
Ana Petrovska, Ioannis Krontiris, Theo Dimitrakos (HUAWEI)
Frank Kargl, Benjamin Erb, Artur Hermann, Nataša Trkulja (UULM)
Guillaume Mockly, Antio Kung (Trialog)
Anderson Ramon Ferraz de Lucena, Alexander Kiening (DENSO)
Francesca Bassi, Ines Ben Jemaa (IRTSX)

Disclaimer

Executive Summary

This deliverable follows up on Deliverable 2.1 “Operational Landscape, Requirements and Reference Architecture – Initial Version” and Deliverable 3.1 “Architectural Specification of CONNECT Trust Assessment Framework, Operation and Interaction” which outlined the core ideas of CONNECT and its trust assessment framework. In particular, D2.1 provided the overall architecture of CONNECT which includes the Trust Assessment Framework (short TAF) as one of its main components. D3.1 then described our three step approach to designing and implementing the TAF.

In particular, we defined the following three steps:

Step 1: Standalone TAF A standalone TAF can operate, for example, within a vehicle or a MEC. While it is able to evaluate trust also on remote entities and data, it will do so solely from its own perspective and based on internal assessment and will not cooperate with other TAFs.

Step 2: Federated TAFs In this step, we add the capability to federate and exchange information among TAFs. This includes exchange of trust models and opinions and establishing an overall understanding of a combined trust model within a large CCAM “Systems-of-Systems”.

Step 3: Digital Twin for Trust Assessment In the final step, we will enable vehicles (or other nodes) to outsource its trust model and TAF state to an external party (typically a MEC) where reasoning on trust models can be executed on behalf of the original TAF.

Besides many foundational aspects for Trust Assessment and the TAF in general, D3.1 described in particular the high-level concept for each of these steps and a high-level architecture for step one, the standalone TAF.

The main contributions in this deliverable are:

1. A fine-grained architecture description of the standalone TAF.
2. A high-level architecture description for the federated TAF.
3. Initial considerations and architectural concepts for the TAF-DT, i.e., a digital twin of a standalone TAF.
4. A detailed specification of the CONNECT Risk Assessment Framework which embeds the TAF into a larger framework that can provide trust evidence to the TAF.
5. Descriptions of two Trust Sources, one based on a Misbehavior Detection Framework, and one based on assessment of available security controls.
6. An approach to calculate the Required Trustworthiness Level (RTL) based on a TARA.

Together with D3.1, this deliverable presents all necessary details underlying our prototype for a standalone TAF. The final version of this document, presented as D3.3 will then provide a more detailed and fine-grained description of the federated TAF and TAF-DT.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Scope and Purpose

This deliverable follows up on Deliverable 2.1 “Operational Landscape, Requirements and Reference Architecture – Initial Version” [13] and Deliverable 3.1 “Architectural Specification of CONNECT Trust Assessment Framework, Operation and Interaction” [10] which outlined the core ideas of CONNECT and its trust assessment framework. In particular, D2.1 provided the overall architecture of CONNECT which includes the Trust Assessment Framework (short TAF) as one of its main components. D3.1 then described our three step approach to designing and implementing the TAF. In particular, we defined the following three steps:

Step 1: Standalone TAF A standalone TAF can operate, for example, within a vehicle or a MEC. While it is able to evaluate trust also on remote entities and data, it will do so solely from its own perspective and based on internal assessment and will not cooperate with other TAFs.

Step 2: Federated TAFs In this step, we add the capability to federate and exchange information among TAFs. This includes exchange of trust models and opinions and establishing an overall understanding of a combined trust model within a large CCAM “Systems-of-Systems”.

Step 3: Digital Twin for Trust Assessment In the final step, we will enable vehicles (or other nodes) to outsource its trust model and TAF state to an external party (typically a MEC) where reasoning on trust models can be executed on behalf of the original TAF.

The main contributions in this deliverable are:

1. A fine-grained architecture description of the standalone TAF.
2. A high-level architecture description for the federated TAF.
3. Initial considerations and architectural concepts for the TAF-DT, i.e., a digital twin of a standalone TAF.
4. A detailed specification of the CONNECT Risk Assessment Framework which embeds the TAF into a larger framework that can provide trust evidence to the TAF.

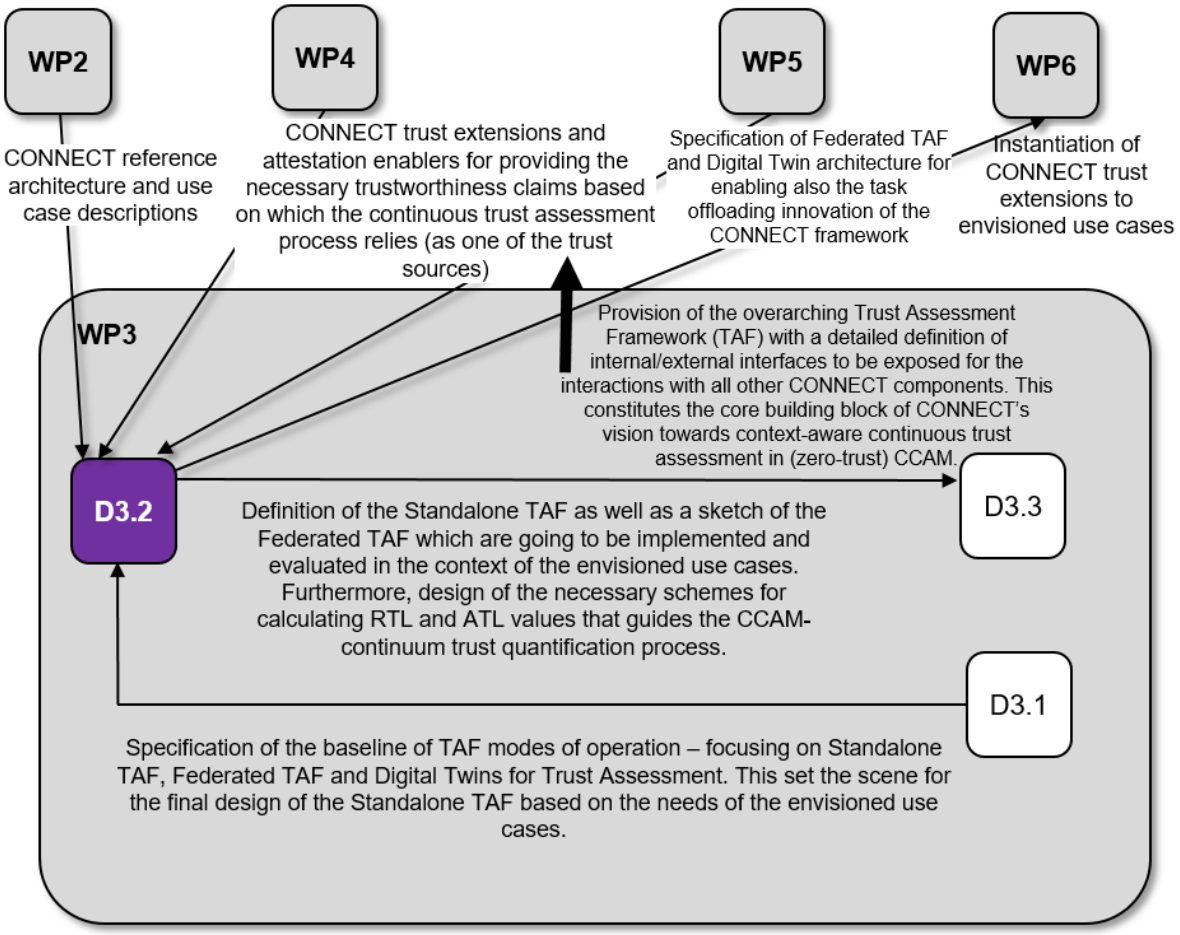


Figure 1.1: Relation of D3.2 with other WPs and Deliverables.

5. Descriptions of two Trust Sources, one based on a Misbehavior Detection Framework, and one based on assessment of available security controls.

6. An approach to calculate the Required Trustworthiness Level (RTL) based on a TARA.

Besides many foundational aspects for Trust Assessment and the TAF in general, D3.1 described in particular the high-level concept for each of these steps and a high-level architecture for step one, the standalone TAF.

1.2 Relationship with other WPs & Deliverables

With the documentation of the research road-map towards the design of a holistic Trust Assessment Framework (TAF), capable of quantifying the trustworthiness of individual CCAM entities (as detailed in D3.1 [10]), this deliverable proceeds with the first version of the Standalone TAF documenting in detail all of its internal building blocks, operations as well as interfaces to be exposed for enabling the interactions with the other CONNECT framework components. *We have to highlight that this constitutes the first complete version of the overarching CONNECT Trust Assessment Architecture which may be subject to refinements as we progress with the implementation, integration and evaluation activities in the context of the envisioned use cases (WP6).* Final version of all TAF variants will be presented in Deliverable D3.3.

In this context, Figure 1.1 depicts the direct and indirect relationships to other tasks and Work Packages (WPs). First, it distills the mechanisms based on which the **Required Trust Level (RTL) and Actual Trust Level (ATL) are defined** for guiding the later **trust quantification process across the entire CCAM continuum**. Recall that CONNECT extends the stand-alone vehicle domain to safe and security solutions distributed from Vehicle to MEC and Cloud facilities so as to support the envisioned automation of *connected vehicles*: This vision is enabled by the vehicle's communication with other entities formulating the Vehicle-to-everything (V2X) landscape. The resulting "*CCAM continuum*" paradigm seeks to seamlessly and securely combine the available hardware and software (from the in-vehicle sensors to the MEC virtualized infrastructure) to support the deployment and operation of certified (using ISO/SAE 21434 [9]) CCAM functions. Whereas the collective consideration (treatment) of the continuum resources presents opportunities for increased performance and a higher degree of automation, the individual Software (SW) and Hardware (HW) infrastructure may exhibit diverse yet dynamic trust states; and when those infrastructures are brought together to make-up the continuum strong trust assessment mechanisms need to also be deployed for asserting to the correct state of all these functional assets.

This is enabled through the design of the first variant of the TAF (**Standalone TAF** capable of analysing the trustworthiness levels of entities in isolation) coupled with the (preliminary) definition of the necessary interfaces that need to be exposed for accommodating the interactions with the other CONNECT components. Especially, as it pertains to the trusted computing capabilities (offered by the CONNECT Trusted Computing Base developed in the context of WP4) towards the runtime integrity verification of all (SW and HW) elements comprising a CCAM function chain to be assessed. This functionality, together with those libraries offered for secure software upgrade and/or state migration (as reaction policies to the change of a node's trust level), comprise CONNECT's TCB capabilities for supporting the operations of the Trust Assessment Framework: *All attestation attributes constitute one of the trustworthiness sources based on which the TAF (as the core WP3 artefact) calculates the Actual Trust Level (ATL) of a (SW and/or HW) element which leads to the trust state characterisation of the target node*.

Overall, the outcome of this deliverable serves as the first implementation milestone of CONNECT's efforts towards the design of an overarching trust assessment framework by consolidating the core architecture of the Standalone TAF ben able to cope with the complex set of trust relationships that need to be assessed in the CCAM ecosystem (based on our envisioned use cases). All these capabilities set the scene for the detailed experimentation and evaluation activities of all these trust extensions in the context of the envisioned use cases on Intersection Movement Assistance, Collaborative Cruise Control and Slow-Moving Traffic Detection (WP6).

1.3 Deliverable Structure

This deliverable, D3.2 "CONNECT. Connect trust & risk assessment and cad twinning framework (initial version)", follows-up on this with a detailed, fine-grained architecture of the standalone TAF as presented in Chapter 4. This fine-grained architecture addresses details left open in D3.1 and also serves as a reference for the prototype implementation of the standalone TAF.

As laid out in D3.1, this deliverable also describes a high-level architecture for the next extension or Step 2, the federated TAF. As described in Chapter /refch:standalone-taf, the federated TAF will enable two TAFs to interact with each other. We distinguish three forms of such an interaction: an interaction where one TAF requests a single trust source in a remote TAF (case 1), sending of

a TAR to retrieve an ATL from by one standalone TAF to another standalone TAF (case 2) and, finally, the case where two TAFs interact to jointly calculate an ATL where trust relationships span linked trust models residing in different TAFs (case 3).

For the federated TAF, Deliverable 3.3 “Connect trust & risk assessment and cad twinning framework (final version)” will then describe the detailed architecture including the functionality integrated into our TAF prototype. We envision that our prototype will support case 1, potentially case 2, but that case 3 will not be implemented due to its very high complexity.

Next, Chapter 6 describes first steps towards the Digital Twin version of the TAF (short TAF-DT) including a high-level architecture of a generic Digital Twin architecture. Again, more details on the architecture will be given in D3.3.

Here, we define what a TAF-DT actually is, and define requirements and challenges related to realizing a TAF-DT. In particular, we identify challenges related to state synchronization and how this creates challenges for consistency, availability, and partition tolerance (CAP) of such a distributed system. CONNECT will implement a subset of the TAF-DT functionality required for evaluation of its feasibility and a trade-off between strong consistency and synchronization on the one hand and performance and latency on the other.

This deliverable then continuous in Chapter 7 with the Risk Assessment Framework which is a major provider of trustworthiness evidence and also structural information to build trust models to the TAFs, detailing the risk assessment methodology and design of the risk assessment engine which can automate analysis of risks in a system. This is also illustrated through a running example.

Chapter 8 continues with a discussion of evidence-based trust opinion calculation. This identifies how atomic trust opinions can be calculated based on either misbehavior detection system or on knowledge about security controls. The particular challenge here is the translation of heterogeneous trust evidence into subjective logic opinions accessible to the TAF. Both cases are implemented as trust sources in our TAF prototype and serve as examples on how to establish future new trust sources.

With the previous chapters in place, we have described all required elements for calculation of Actual Trustworthiness Levels (ATLs). What is missing is the calculation of Required Trustworthiness Levels (RTLs). This is considered in Chapter 9 where an approach is presented which calculates the RTL based on risk analysis through a TARA.

Chapter 10 concludes this initial version of the deliverable. An updated presenting a substantial update on all aspects of the TAF and WP3 results will be provided by D3.3 due on M30 of the project.

Chapter 2

Trust Assessment Terms and Definitions

This chapter summarizes terms and definitions related to trust assessment in general, as well as the the Trust Assessment Framework (TAF) itself. This vocabulary will act as the reference resource, throughout all project activities, so that all stakeholders can have a common understanding of the characteristics that could be used to describe the trustworthiness of a data item or node.

While this vocabulary primarily targets documentation of terms capturing the mode of operation of a TAF in the context of Autonomous Vehicles (AVs), it is intended for use horizontally in the information technology domain and all domains where Subjective Logic is used as the foundation of trust reasoning.

ATL (Actual Trustworthiness Level) The ATL reflects the result of an evaluation of a specific (atomic or complex) proposition for a specific scope provided by the TLEE. It quantifies the extent to which a certain node or data can be considered trustworthy based on the available evidence.

ATO (Atomic Trust Opinion) An ATO is a subjective logic opinion created by the TSM when quantifying one specific type of a Trust Source based on trustworthiness evidence. An ATO is formed by a Trust Source in the context of a single trust relationship.

PROP (Proposition) A *proposition* is a logic statement about some phenomenon of interest whose level of trustworthiness we are interested in assessing. A proposition could be 1) atomic—a proposition whose truth or trustworthiness can be directly assessed, or 2) composite, comprising of multiple atomic propositions. The proposition describes the fulfillment of the properties in relation to *data* or *nodes*.

RTL (Required Trustworthiness Level) The RTL reflects the amount of trustworthiness of a node or data that an application considers required in order to characterize this object as trusted and rely on its output during its execution.

TA (Trust Assessment Manager) The TAM is a component inside the TAF which orchestrates the overall process of trust assessment.

TAF (Trust Assessment Framework) A software framework which, given a trust model for a specific function running inside a CCAM system, is able to evaluate trust sources for trustworthiness evidence and evaluate propositions within the trust model to obtain their ATLs.

Optionally, also an RTL can be evaluated and trust decisions can be taken and communicated to the application.

TAF-API (TAF - Application Programming Interface) Application Programming Interface by which the TAF and its functionality can be accessed from an application.

TAF-DT (TAF - Digital Twin) The digital twin allows a vehicle to replicate its TAF including among others its TMs and TSs within a MEC. This allows a vehicle to outsource trust assessment to a MEC where the TAF-DT is expected to run inside a TEE so that confidentiality and integrity of its data and state can be protected from the MEC.

TDE (Trust Decision Engine) The TDE is a component inside the TAF which performs the last step before an output is provided to the application that requested trustworthiness assessment. The TDE either forwards the Actual Trustworthiness Level (ATL) calculated by the TLEE along to the application or outputs a Trust Decision (TD). A TD is created after comparing the ATL to the Required Trust Level (RTL) in a predetermined manner. Whether the output of the TAF is an ATL or a TD depends on the needs of the application requesting trustworthiness assessment.

TLEE (Trustworthiness Level Expression Engine) The TLEE is a component inside the TAF that calculates the level of trustworthiness for a concrete trust model instance and the proposition that needs to be evaluated. The TLEE uses the numerical values of the Atomic Trust Opinions computed based on the trust sources collected in the TSM. Based on these inputs, the TLEE calculates an ATL and provides it to the TA. The TLEE encapsulates most of the Subjective Logic formalism.

TM (Trust Model) Trust model is a graph-based model which is built on top of a system model which represents all components and data needed to perform a certain function. Components represented either create, transmit, process, relay, and receive the data used as input to a function. The vertices in a trust model correspond to an abstraction called trust objects, and the edges in a trust model correspond to trust relationships between a pair of trust objects. The trust model also encompasses a list of trust sources used to build up / quantify trust relationships by providing atomic trust opinions. The trust model is a main input to the TMM and the TLEE. Since trust is a directional relationship between two trust objects and it is always in relation to a concrete property or scope, then as part of the trust model, there can be multiple trust relationships between the same two trust objects, depending on different properties of the trust relationship, or the scope of the trust relationship.

TMM (Trust Model Manager) The TMM is a component inside the TAF responsible for storing trust models and making them accessible for TLEE and other purposes. In particular, it is able to provide TMs for specific functions running in a CCAM system, also considering different scopes that TMs may cover.

TO (Trust Opinion) The Trust Opinion is a numeric value which represents the trustworthiness of an entity as assessed by another entity based on relevant entity. Within the scope of CONNECT a Trust Opinion is expressed in form of a Subjective Logic binomial opinion ω_B^A where A is the entity assessing trustworthiness and B is the entity whose trustworthiness is being assessed..

Trust Objects Trust objects are core building blocks of a trust model. They represent entities that assess trust or for which trust is assessed. The trust objects are identified 1) based

on the *components* from the component diagram and 2) the *atomic propositions* (i.e., the properties about data or nodes for which trust assessment is conducted).

Trust Relationships A trust relationship is a directional relationship between two trust objects that are called trustor (i.e.,, the “thinking entity”, the assessor) and a trustee (one who is trusted). The trust relationship is always in relation to a concrete property and a certain scope.

Trustworthiness Tier Trustworthiness Tier is a categorization of the levels of trustworthiness which may be assigned by the TAF (or another Verifier that appraises the attestation results of a TC and communicates them to the TAF) to a specific Trustworthiness Claim.

Trustworthiness Claims A Trustworthiness Claim (TC) is a form of node-centric ATO provided by a TS and contains a specific data quote used for conveying the information needed by the TAF to make a decision on the trust level of an object. The TC is usually produced (by the Attester) so as to provide trustworthiness evidence (cf. “Trust Source”) that can be used for appraising the trustworthiness level of the Attester in a *measurable* and *verifiable* manner [**?**]. Measurable reflects the ability of the TAF to assess an attribute of the Attester against a pre-defined metric while verifiability highlights the need for all claims to have integrity, freshness and to be provably & non-reputably bound to the identity of the original Attester. Examples sets of TCs might include (among other attributes) evidence on system properties including: (i) **integrity** in the context that all transited devices (e.g., ECUs) have booted with known hardware and firmware; (ii) **safety** meaning that all transited devices are from a set of vendors and are running certified software applications containing the latest patches and (iii) **communication integrity**. For a more detailed list of possible system (behavioural) evidence that can be appraised as part of TCs, please refer to Section **??**.

TS (Trust Source) A TS manages one or multiple trustworthiness evidence inside the TAF. On request of the TA, it quantifies the trustworthiness of a trustee based on a specific type of evidence in form of an atomic trust opinion.

TSM (Trust Source Manager) The TSM is a component inside the TAF responsible for handling all available TS inside a TAF and to establish and integrate new TSs dynamically through a plugin interface.

Chapter 3

Running Example

This chapter provides a use-case that will be referenced to in some of the upcoming chapters, where it is useful to illustrate the described approach with a specific example.

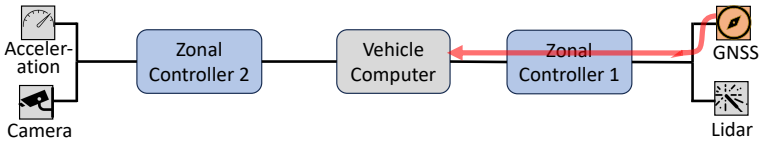


Figure 3.1: Simplified in-vehicle network use case

The Figure 3.1 shows the use-case in a simplified in-vehicle architecture. The main objective is to show the path taken by GNSS data through the vehicular network. The GNSS data, carrying the vehicular position as main information, is received by the GNSS ECU and forwarded to the Vehicle Computer passing through the Zonal Controller 1. Zonal Controller 2, Camera, and Acceleration ECU are part of the vehicular architecture, but not part of the scope of this example. All components in this network are nodes that can create or process data and are therefore able to compromise the data. We assume that a CCAM application is running in the Vehicle Computer, such as a CACC application. The application wants to assess the trustworthiness of the position data originally provided by the GNSS ECU but forwarded by multiple intermediary entities. Based on the trustworthiness of the position data, the application decides whether or not to use the received position.

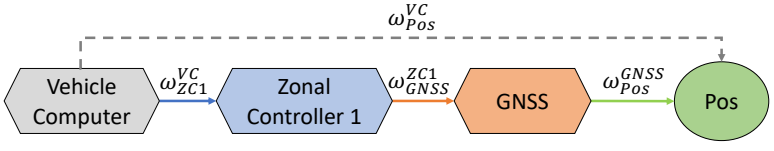


Figure 3.2: Trust Model derived from the in-vehicle network.

Based on the in-vehicle network, a Trust Model can be derived. Such a simple Trust Model is shown in Figure 3.2. In the Trust Model, all data items, i.e., the position and the entities creating and forwarding the position, are represented as nodes. As shown in Figure 3.2, the Trust Model has three trust relationships in the trust chain (ω^{VC}_{ZC1}, ω^{ZC1}_{GNSS}, and ω^{GNSS}_{Pos}). The final goal is to create the opinion ω^{VC}_{Pos} that the VC has on the position even though the VC does not observe the position directly (i.e., it does not have a direct relationship with the position). This is done by discounting the calculated opinions on the three trust relationships in the trust chain ($\omega^{VC}_{Pos} = \omega^{VC}_{ZC1} \otimes \omega^{ZC1}_{GNSS} \otimes \omega^{GNSS}_{Pos}$).

Chapter 4

Standalone TAF

4.1 Overview

CONNECT's Trust Assessment Framework (TAF) is a software system assessing the level of trustworthiness of entities with respect to a specific scope. That said, the TAF is conceptually a reactive software system that receives incoming messages that provide updates about the trustworthiness of surrounding environment or relevant entities. These updates are then processed and integrated into an internal representation of trust sources, trust opinions, and relationships as part of trust models. At the same time, the TAF provides a service to other applications and entities by providing access to its trustworthiness assessment based on these internal representations and derived information. The generic TAF needs to be deployable onto different runtime environments, in particular onto vehicles for in-vehicular and V2X use cases and onto MEC servers for stationary V2X usage in infrastructure.

The following list of requirements have shaped the architecture and design of the TAF:

Accuracy: The TAF provides critical information on which other components will base decisions that can affect behavior and even safety of vehicles. Therefore, it is of high importance that trust assessments provided are accurately reflecting trustworthiness of components as much as possible to the degree by which evidence is available.

Efficiency: Given the critical functionality of the TAF, it is important that updates received by the TAF are reflected as fast as possible in the trustworthiness assessment calculated the TAF. As the actual worth of assessments substantially depends on the timeliness of their availability, low processing delays are mandatory. At the same time, the runtime environment of the TAF might be resource-constrained so that computational efficiency is vitally important.

Scalability: Both the scale of trust models and the scale of scenarios may vary a lot between different use cases the TAF is applied to. This means that the TAF needs to be able to successfully operate despite of varying rates of incoming messages, varying numbers of applications that use the service of the TAF, varying levels of dynamicity and complexity of individual trust models, and varying amounts of entities that have to be assessed in these models. Hence, scalability is a key requirement for the TAF.

Openness & Extensibility: While the standalone TAF is designed as a single component, the federated use should already be taken into account at design time of the standalone TAF.

This includes services and interfaces in which different TAF instances can exchange and share data to enhance each others trustworthiness assessments. Furthermore, the TAF should easily be adapted to address other use cases of trust assessment in the future. Therefore, the TAF should be easily extendable to incorporate new sources of trust evidence, new trust models, and new applications requiring new assessments.

Configurability & Customizability: Based on different use cases and runtime environments (e.g., vehicles vs. MECs), the TAF requires configuration parameters to allow for certain trade-offs to address the requirements mentioned above. For instance, in case of resource limitations or for use cases with many concurrently occurring entities, a TAF configuration should allow to trade accuracy for timeliness (e.g., by micro-batching updates under high load) or to trade lower response latencies for higher memory consumption (e.g., by extensively caching results). In addition, individual components of the TAF should modularized so they can be swapped easily. This would allow also for customized or optimized components for specific use cases or dedicated runtime environments.

Security: Given the critical functionality of the TAF, it is important that its processing and results are protected in terms of their integrity from malicious manipulation. Furthermore, confidentiality may be a requirement as far as the trust model and values may provide insights into vehicle and systems that should not be public. This is of particular importance if the TAF is running as TAF-DT in a MEC server.

4.2 Architecture

The TAF architecture addresses the requirements mentioned above by applying specific design principles for its internal architecture. To decompose the complex and concurrent tasks to be handled by the TAF, we follow an event-driven design in which dedicated modules process certain events and can in turn emit events for other components.

4.2.1 System Overview

The TAF is internally organized as a modular system consisting of different components, as illustrated in the high-level architecture in Figure 4.1. The *Trust Model Manager* (TMM) provides an internal repository of available trust models and instantiates appropriate trust model instances upon application requests. Furthermore, it manages the full life cycle of all trust model instances by incorporating potential changes. The *Trust Source Manager* (TSM) maintains a list of possible trust sources that could be used as evidence for assessing the trustworthiness of an entity. By continuously assessing evidence from external trust sources, the TSM dynamically incorporates trust opinions into existing trust model instances. The *Trustworthiness Level Expression Engine* (TLEE) takes a trust model instance as an input and assesses the level of trustworthiness of the included propositions. The *Trust Decision Engine* (TDE) eventually forms trustworthiness and trust based upon output from the TLEE. Finally, the *Trust Assessment Manager* (TAM) is a central component that orchestrates updates from the TMM & TSM and schedules TLEE computations and TDE invocations. The TAM also implements external interfaces so that the trust assessment service of the TAF can be accessed by applications. More details about the individual components of the TAF architecture are described in Section 4.5, following a specification of the external services (see Section 4.4).

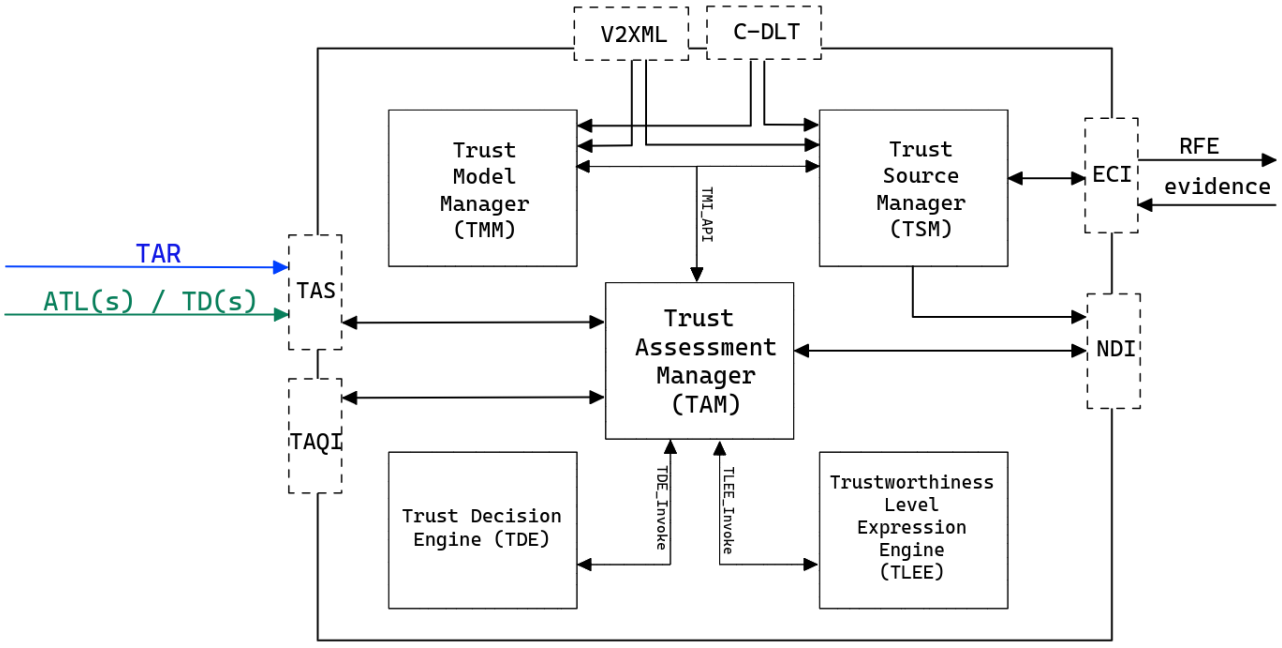


Figure 4.1: High-level architecture of the Trust Assessment Framework (TAF). This includes the five main components: Trust Model Manager, Trust Source Manager, Trust Assessment Manager, Trust Decision Engine, and Trustworthiness Level Expression Engine. The illustration also shows the three main internal interfaces of the TAF, namely the Trust Model Instance API (TMI_API) and the interfaces for the TLEE Invocation (TLEE_Invoke) and the TDE Invocation (TDE_Invoke). In addition, the six external interfaces and services are depicted – the Trust Assessment Service (TAS), the Trust Assessment Query Interface (TAQI), the V2X Message Listener (V2XML), the Connect DLT (C-DLT), the Evidence Collection Interface (ECI), and the Node Discovery Interface (NDI).

4.2.2 Key Design Principles

The basic architecture of the TAF follows a *concurrent, event-driven architecture* in which a number of components await, process, handle, and potentially also emit events. Based on the different external services and interfaces as well as internal tasks the TAF has to integrate, we can separate these components conceptually into the following groups:

(i) Components that process updates from outside the TAF and publish them as TAF-internal events (e.g., V2X Message Listener).

(ii) Components that process internal updates by triggering changes of internal representations (e.g., Trust Model Manager).

(iii) Components that observe changes to internal representations and schedule downstream computations (e.g., Trust Assessment Manager).

(iv) Components that execute dedicated computations upon invocation (e.g., TLEE).

(v) Components that provide services to external entities in order to access internal representations and computation results (e.g., Trust Assessment Manager).

Note that this separation corresponds to an unidirectional flow of execution for the most part: External updates (e.g., CAM messages) are captured and eventually reflected in internal representations (e.g., trust model instances). Internal representations are then used as an input to run computations (TLEE) and store their results (e.g., cache for recent ATLs). Orthogonal to this, the results of recent computations are used to provide services to applications (e.g., respond to TARs, send notifications).

At the same time, the TAF represents a monolithic application that runs as a single application process on a single machine. This choice of architecture and separation of concerns between components allows for the following design goals:

High concurrency: Being designed as a single application process, the TAF requires extensive multi-threading to take advantage of actual parallelism of modern CPU architectures and to provide scalability and performance, even under load. In our event-based architecture, this can easily be done by using dedicated event processors for the different components. Depending on the statefulness and data consistency requirements of components, some components can even employ a pool of threads to handle events concurrently inside the component.

Note that the TAF architecture is not designed to be a distributed application in which multiple TAF instances provide a single, unified service (e.g., by sharding). In particular, this means that no internal events are published to the outside and instead only well-defined services/interfaces are used for external interaction with the TAF.

That said, the TAF does not use an external message queue/broker *between its components* for a number of reasons: (i) additional message latencies due to network messaging (ii) unrealistic messaging abstractions (e.g., having an unbounded message queue between components by potentially offloading messages to disk), and (iii) increased overhead when using additional threads for sending and consuming messages. Instead, the TAF only uses language-internal mechanisms (i.e., shared memory) for disseminating events internally.

Low coupling: The TAF architecture provides low coupling between most components. Stronger coupling (i.e., direct function calls) is only used between the Trust Assessment Manager and the TLEE for scheduling computationally expensive operations and for subsequent calls to the TDE. Low coupling for the rest of the TAF is desirable for a number of reasons:

(i) As processing and emitting events is the primary mode of interaction between components, components do not need to know much about the internals of other components. Instead, components only need to know potential event types and can implement reactive behavior to these events. New event types can be defined without interfering with existing components and implementations can be switched transparently as long as required event handlers are implemented.

(ii) The separation of components into independent event processors backed by threads or thread pools allows for bulk-heading behavior in case of load peaks, which is more favorable than back-pressuring behavior given the responsibilities of the TAF. When using back-pressure, an overloaded component slow downs upstream components and eventually slows down the system's ability to process any additional messages from the outside. Such a behavior is important if a system requires guaranteed processing of each single message and needs to throttle down an external source. However, in our scenario, we have no control over the amount and volume of external

messages. Even further, we would deliberately drop older messages over newer messages for processing if necessary, as newer messages usually contain a more recent update from the outside world. This type of load-shedding helps to keep the service running with reasonable latency. With bulk-heading, we can isolate components in a way that an overloaded component has minimal impact on other components. For instance, this would allow the TAM to invoke the TLEE and the TLEE to run its computations even if the Trust Source Manager is under high contention due to an unexpected surge of messages.

Single Writer Principle: Conceptually, the TAF could be considered to be a highly concurrent in-memory database that maintains trust model instances and associated data items and eventually runs calculations on subsets of its data. Over their life cycle, most trust model instances will experience phases with very high update rates. Given the potentially large number of concurrent trust model instances, as well as the high number of concurrent modifications to these instances, it is important to be aware of potential bottlenecks. Our approach to allow for high levels of concurrency is (i) to reduce the amount of shared state and usage of concurrent data structures whenever shared state is necessary (ii) to apply the single writer principle whenever possible. In this principle, only a single execution unit is allowed to modify a data structure while others have read-only access. This principle prevents write contention between multiple writers and is also aligned with implicit optimizations of modern CPU architectures (i.e., caching). Similarly, we consider ownership as a concept to be used between the TAM and the TLEE. The TAM derives a copy of the structure and the values from the latest state of a trust model instance to be used for an TLEE computation. These copies are then passed to the TLEE as parameters and will not be accessed or modified by the rest of the TAF anymore. Similarly, the resulting data structure returned from the TLEE is considered to be immutable.

Pluggable Modules: TAF components that interact with the outside world (e.g., V2X Message Listener, CONNECT DLT) can be swapped for mockup modules. This facilitates development and testing, but it also allows us to use specific adapters when running emulations or workload generators.

4.3 Trust Model Implementation

In the CONNECT Deliverable 3.1 [10], we had defined a trust model as:

> *A trust model is a graph-based model which represents all components and data needed to perform a certain function. It consists of trust objects and directional trust relationships between trust objects (i.e., the trust network). It also stores trust sources used to quantify trust relationships.*

This is a relatively generic and broad definition which we refine in the following to better capture the details and implementation of trust models in CONNECT.

Therefore, we extend the trust model definition in a way that focuses on the trust model's functionality allowing us to better capture the aforementioned.

At its core, a Trust Model is a directed acyclic graph data structure whose nodes we refer to as **Trust Objects**. A visual concept of a Trust Model is shown in Figure 4.2.

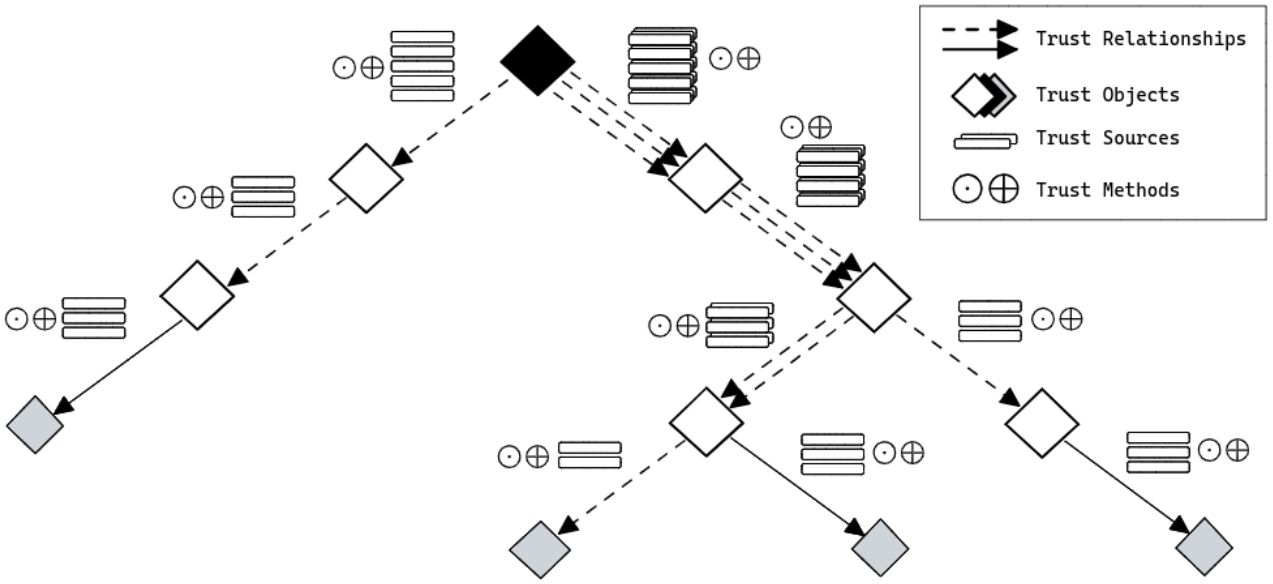


Figure 4.2: A visual concept of a Trust Model.

A Trust Model always has a single node that has no incoming edges which we refer to as the *root node* of the Trust Model. The root node represents the entity for which the trustworthiness is being assessed which we refer to as the **Agent** (the black Trust Object in Figure 4.2). The Trust Model also has at least one leaf node representing the entity whose trustworthiness is being assessed which we refer to as the **Proposition** (the grey Trust Objects in Figure 4.2). There are at times intermediary nodes between the root node and the leaf nodes as shown in Figure 4.2 (the white Trust Objects).

The arrows in the visual concept shown in the Figure 4.2 represent that there is a **Trust Relationship** between trust objects, where the source node is referred to as a *Trustor* and the target node is referred to as a*Trustee*. A Trust Relationship was defined in Section 4.1 of Deliverable D3.1 [10] and is expressed in form of a subjective logic binomial opinion, $\omega_{Trustee}^{Trustor}$, as defined in Section 5.4.1 of Deliverable D3.1 [10].

The **Trust Sources** embedded in the Trust Model represent the entities which the Trust Assessment Framework needs to collect evidence from to quantify and assess a single Trust Relationship. There is a set of Trust Sources attached to every single Trust Relationship inside a Trust Model, but their number and type will vary among Trust Relationships.

The **Trust Methods** are also embedded in the Trust Model as they are necessary to either quantify a set of evidence from a single Trust Source to an Atomic Trust Opinion, to fuse Atomic Trust Opinions into a Trust Opinion, or to fuse two or more Trust Opinions into an Actual Trustworthiness Level.

Moreover, the functionality of a trust model could be generalized as follows:

(i) It allows to *specify* all information at *design-time* which are relevant for trustworthiness assessment and are used by the TAF at run-time.

(ii) It allows to *store* quantifications oftrustworthiness at different stages of trustworthiness assessment during TAF's *run-time*.

Specifically, a Trust Model:

1. *Specifies* the **Trust Objects** whose trustworthiness should be assessed with the goal of evaluating the **Actual Trustworthiness Level(s)** for a specific application or function.
2. *Specifies* the **Trust Relationships** representing Trust Opinions between any two Trust Objects which need to be quantified.
3. *Specifies* the **Trust Model Instantiation Policies** for instantiating additional Trust Objects inside a single Trust Model Instance at run-time when needed. A Trust Model will change its structure and incorporate, for example, new trust objects if a new vehicle appears in a cooperative driving scenario.
4. *Specifies* a list of all of the **Trust Sources** whose evidence is needed for assessing the trustworthiness of Trust Objects in form of a **Trust Opinion**.
5. *Specifies* mathematical Trust Methods for fusing Atomic Trust Opinions to create **Trust Opinions**.
6. *Specifies* a mathematical method for fusing two or more **Trust Opinions**, when needed to produce an **Actual Trustworthiness Level**.
7. *Stores* all of the assessed **(Atomic) Trust Opinions** for each individual Trust Relationship.
8. *Stores* all of the assessed **Actual Trustworthiness Levels**.

A trust model at design time contains specifications of the potential structure of the model and how the trust model can be instantiated, populated, and maintained at runtime.

4.3.1 Trust Model Templates and Instances

To be able to distinguish between a design-time trust model and a run-time trust model, we defined the following two manifestations of trust models – templates and instances:

Trust Model Template (TMT) is a design-time manifestation of a trust model. A Trust Model Template is created at design-time with the goal of capturing all of information necessary for the Trust Assessment Framework to assess trustworthiness for a specific application upon the receipt of a Trustworthiness Assessment Request. A Trust Model Template specifies all of the relevant Trust Objects, Trust Model Instantiation Policies, Trust Relationships, Trust Sources, and Trust Methods which are to be instantiated as part of a Trust Model Instance. As such, a uniquely identifiable, application-specific Trust Model Template is used by the Trust Model Manager to instantiate a corresponding Trust Model Instance at run-time, making a TMI completely dependent on the design of the TMT.

Trust Model Instance (TMI) is a run-time manifestation of a trust model. A Trust Model Instance reflects the TAF's current and latest view on the world and is thus a single, internal source of truth for any computations to be done during trust assessment. The TAF uses the information inside a Trust Model Instance to know which Trust Sources to instantiate, which Trust Relationships need to be quantified in form of a Trust Opinion, and which Trust Opinions form an ATL. However, a TMI is also designed to serve as a data structure to store Atomic Trust Opinions, Trust Opinions, and Actual Trustworthiness Levels during the lifetime of an instance. As such, a TMI will be continually updated with recalculated opinions and levels.

Trust model templates can specify varying levels of dynamicity regarding the topology used by the model at runtime. Static trust model instances have a fixed topology that is already fully determined at design time in their template. In turn, there are also trust model instances whose graph structure can change over time based on the policies inside their corresponding template **and** external input to the TAF, such as CAMs and CPMs. We refer to the prior as **Static Trust Model Instances** and the latter as **Dynamic Trust Model Instances**.

Static Trust Models Instances

The structure of Static Trust Model Instances is solely determined by their Trust Model Templates. The graph structure of the static Trust Model Instance is supposed to stay the same throughout the lifetime of the instance and it is not supposed to change based on any input received during run-time.

In case of in-vehicular applications which utilize the in-vehicle network to collect input data, like in our running example, the graph structure of the Trust Model Instance is identical to the predefined structure specified in the corresponding Trust Model Template. Its structure is static in the sense that it does not change over time, i.e., all of the different Trust Relationships and the Trust Objects stay the same. This is because static Trust Models Instances are instantiated for applications whose data flow is fixed and for which the nodes the data flows through are known in advance. However, different types of trustworthiness opinions such as Atomic Trust Opinions, Trust Opinions and ATLs will be stored inside the instance at run-time and they are expected to change over time depending on changes in evidence received.

An example Trust Model Template for the use-case of Adaptive Cruise Control (ACC) is shown in Figure 4.3. As can be seen from the figure, the TMT consists of Trust Objects, Trust Relationships, Trust Methods, and Trust Sources. The TMT has a single root node, VC, representing the Vehicular Computer, which is the Agent for which the TAF would be assessing trustworthiness as this is where the ACC application will be running. Moreover, there are four leaf nodes, R_{data}, G, C_{data}, and L_{data}, representing the radar data, the GNSS sensor, the camera data, and the LIDAR data, which are the propositions, i.e., the entities whose trustworthiness we need to assess in form of Actual Trustworthiness Levels. In order to do that, we need to first quantify every single relationship which is represented by a directed edge, including the relationships between the Vehicle Computer Trust Object, VC, and the Zonal Controller Trust Objects, ZC5 and ZC2. This will be done by analyzing evidence from pre-determined Trust Sources and by using pre-selected Trust Methods to perform the quantification of evidence, all of which have already been specified

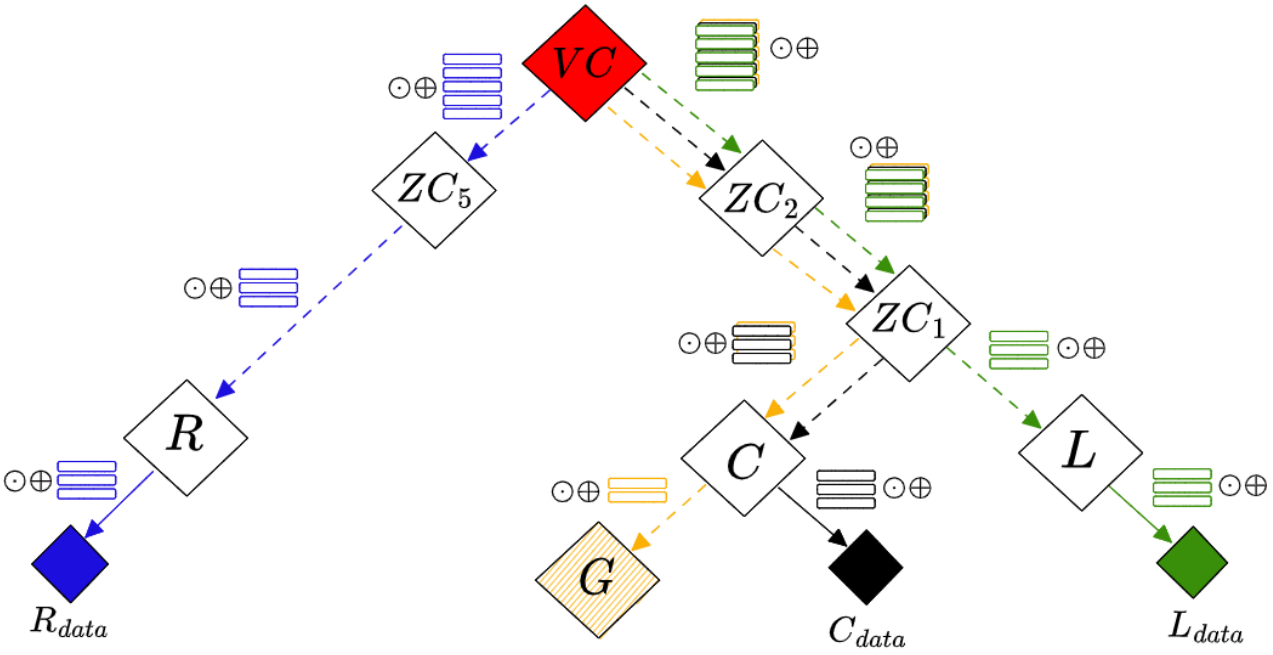


Figure 4.3: A visual concept of a Trust Model Template - static use-case.

in the TMT. But before all of this is done at run-time, a Trust Model Template needs to be instantiated to create a Trust Model Instance.

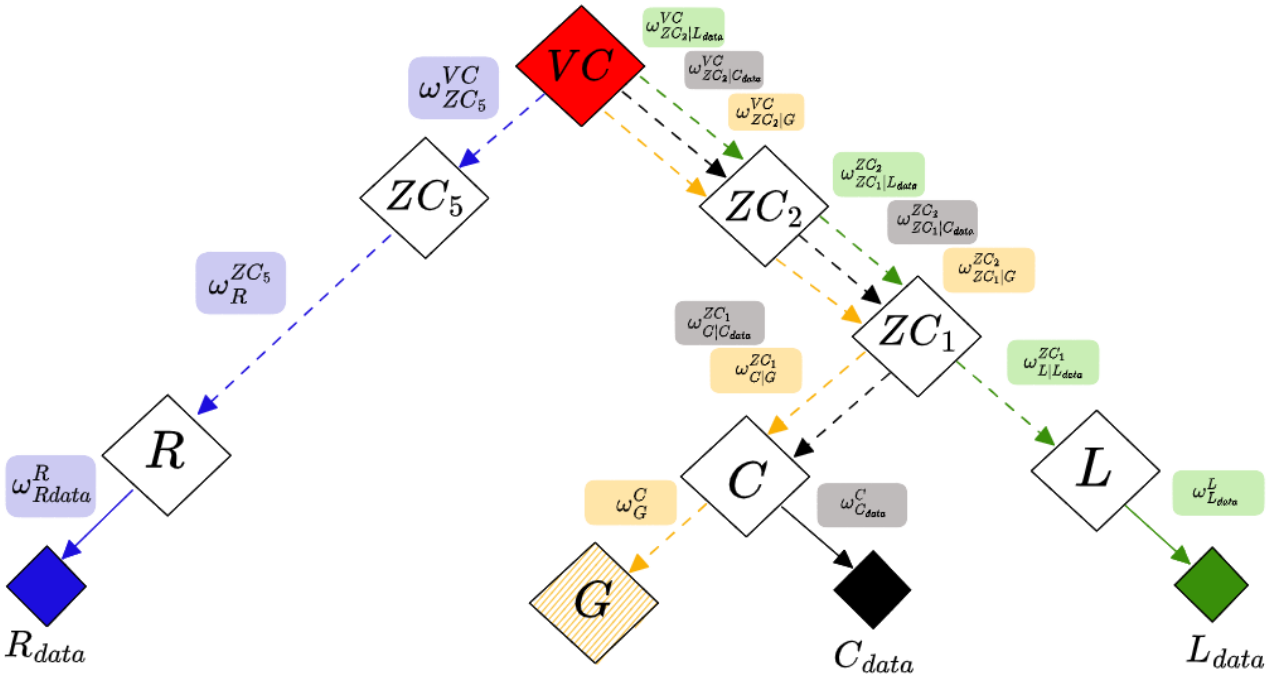


Figure 4.4: A visual concept of a static Trust Model Instance

An example of a Static Trust Model Instance for the use-case of Adaptive Cruise Control is shown in Figure 4.4. As can be seen from the figure, the structure of the TMI is the same as that of its corresponding TMT in Figure 4.3. What is different, however, is that the instance will at some point contain numerical values which have been cached into it in the process of trustworthiness

assessment. The figure shows only Trust Opinions between relevant Trust Objects, such as $\omega^{VC}_{ZC_5}$, $\omega^{L}_{L_{data}}$, etc. for the sake of not overloading the figure, but the TMI that the figure represents is intended to also cache the Atomic Trust Opinions that the TSM uses to create the Trust Opinions, as well as the ATLs which the TLEE forms from Trust Opinions through the process of trust discounting.

Note that for applications which are represented by Static TMIs, a single TMI is needed to assess the trustworthiness of all the of relevant input data. That is, a single static TMI per application is enough to calculate all of the necessary propositions. This is not the case for Dynamic Trust Model Instances.

Dynamic Trust Models Instances

Dynamic Trust Model Instances are instantiated based on their Trust Model Templates **and** based on external input received by the TAF, such as Cooperative Awareness Messages (CAMs) and Collective Perception Messages (CPMs). Unlike in the case of Static TMIs, there can be multiple Dynamic TMIs per single application i.e. multiple Dynamic TMIs all based on a single TMT. This is due to the fact that Dynamic TMIs are instantiated per sender of a CAM or a CPM, and each sender that sends a CAM/CPM to the ego vehicle running the TAF will have its own corresponding TMI inside the TAF.

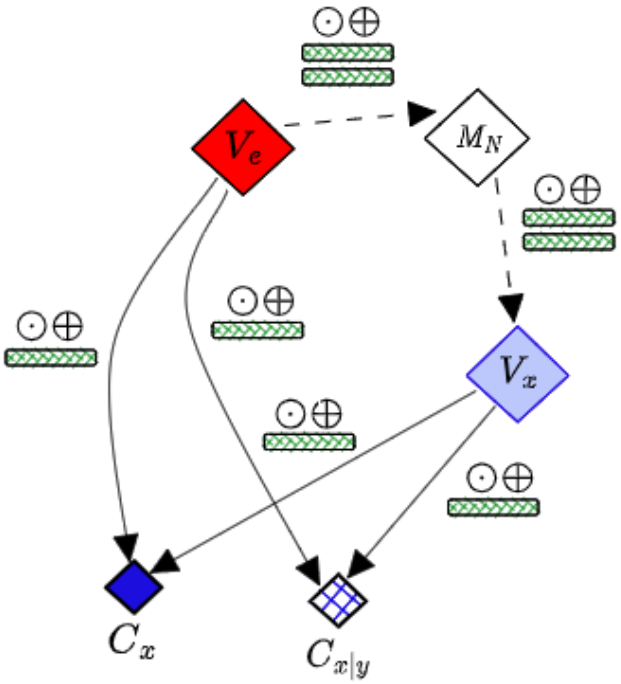


Figure 4.5: A visual concept of a Trust Model Template - dynamic use-case.

An example of a Trust Model Template used for the use-case of a Local Dynamic Map (LDM) which would store trustworthiness of vehicular information extracted from CAMs and CPMs is shown in Figure 4.5. As can be seen from the figure, this TMT also consists of Trust Objects, Trust Relationships, Trust Methods, and Trust Sources. The TMT has a single root node, V_e,

representing the ego vehicle, which is the Agent for which the TAF would be assessing trustworthiness. Moreover, there are two leaf nodes in this TMT, C_x, representing a CAM message sent by vehicle V_x, and $C_{x|y}$, representing a CPM message sent by vehicle V_x containing information about vehicle V_y. These are the propositions, i.e., the entities whose trustworthiness the TAF needs to assess in form of Actual Trustworthiness Levels. In order to do that, the TAF would build a Trust Opinion on C_x and $C_{x|y}$ directly. However, this TMT suggests that a secondary opinion on the received CAM and CPM should be built by collecting an opinion of a Mobile Edge Computer (MEC), M_N, on the vehicle V_x which sent the messages. There is also a relationship between the V_x and its own messages, as well as a relationship between the ego vehicle, V_e, and the M_N that provided an opinion on the vehicle which sent the messages V_x. But before all these relationships are assessed at run-time, this Trust Model Template would need to be instantiated to create a Dynamic Trust Model Instance.

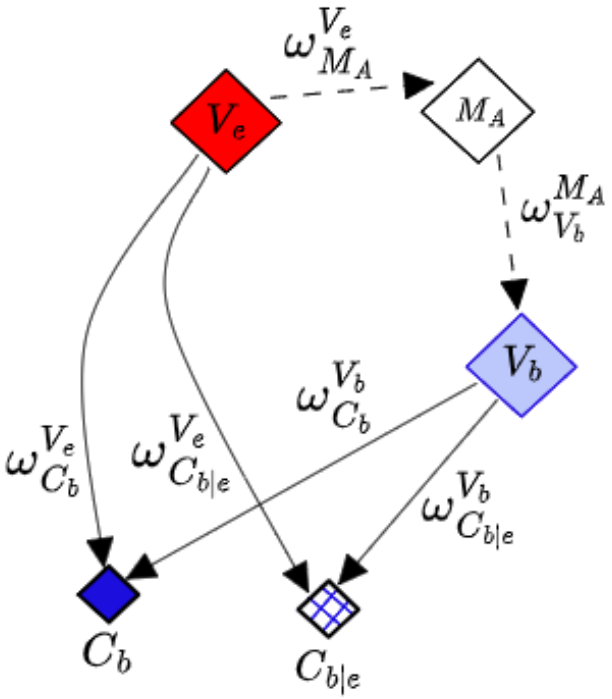


Figure 4.6: A visual concept of a Dynamic Trust Model Instance

An example of a Dynamic Trust Model Instance for the use-case of a Local Dynamic Map is shown in Figure 4.6. As can be seen from the figure, the graph structure of the initial TMI is the same as the fixed structure specified in its corresponding TMT in Figure 4.5. However, Trust Objects have been instantiated for specific entities. For example, a Trust Object has been instantiated for a specific vehicle, V_b, which sent a CAM, C_b, containing information, such as speed and position, about itself, and a CPM, $C_{b|e}$, containing information about the ego vehicle. Trust Objects have been instantiated for these two messages as well in this TMI. Moreover, a Trust Object has been instantiated for a specific Mobile Edge Computer, M_A, which is the MEC that can provide an opinion on the sender vehicle, V_b. As can be noted, a Dynamic Trust Model Instance focuses on a single sender vehicle and the trustworthiness of its messages. Similar to Static TMIs, Dynamic TMIS will cache Atomic Trust Opinions, Trust Opinions, and ATLs at run-time and they are also expected to change over time depending on changes in evidence.

Moreover, the graph structure of the Dynamic Trust Model Instance is expected to change during the lifetime of the instance. This is because Dynamic Trust Model Instances represent vehicular applications which use CAMs and CPMs as input and, as such, require the TAF to calculate trustworthiness of incoming CAMs and CPMs. As CAMs and CPMs are sent about different vehicles at different points in time, the Trust Model Manager needs to dynamically update the structure of a Trust Model Instance to include new Trust Objects and remove the ones which are no longer relevant.

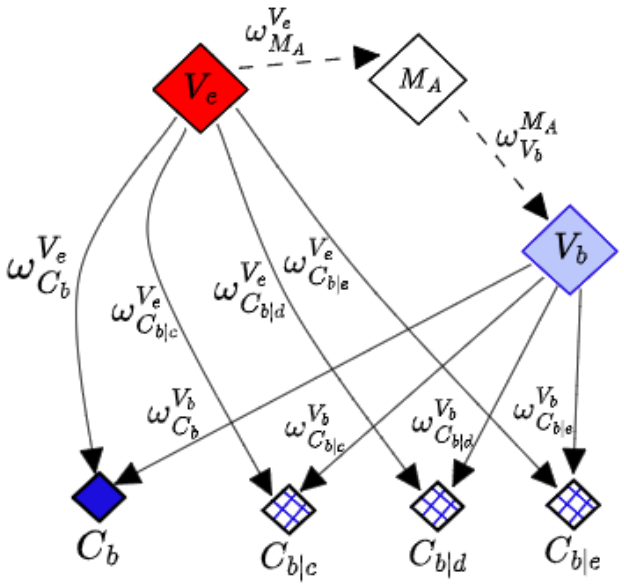


Figure 4.7: A visual concept of a Dynamic Trust Model Instance whose structure changed

An example of what the change of the structure of a Dynamic TMI could look like is shown in Figure 4.7. This example TMI is an extension of the initial TMI shown in Figure 4.6. It would have been extended upon the receipt of a CPM from the same vehicle V_b that contained information about vehicles V_c and V_d. These would be represented as Trust Objects $C_{b|c}$ and $C_{b|d}$ in the updated Dynamic TMI shown in Figure 4.7. Should vehicles V_c and V_d get out of detection range of the sender vehicle V_b, V_b would no longer include information about these vehicles in its CPM messages received by the ego vehicle V_e. This would trigger the deletion of Trust Objects $C_{b|c}$ and $C_{b|d}$ at certain point in time.

The dynamic addition and deletion of Trust Objects to a single Trust Model Instance can result in multiple changes of the structure of the Trust Model Instance over time. Hence, it is important to be able to distinguish between different TMIs at different points in time. Moreover, we anticipate that even the Trust Model Templates may need to be changed over time, albeit significantly less frequently as compared to the Trust Model Instances. For this, we propose to have the Trust Model Versioning scheme as explained in the following subsection.

4.4 External Interfaces & Services

In this section, we specify the system design underlying the standalone TAF prototype being developed within the CONNECT project. To this end, we specify the communication and interaction of the TAF with other entities outside the TAF (e.g., applications and their respective functions, components on the same node, other vehicles, MECs). This includes (i) services that the TAF provides to other entities, (ii) services from other entities that the TAF utilizes for its own internal implementation, (iii) listening interfaces to passively receive notifications from other external components.

4.4.1 Interface & Service Basics

Before we describe each of the external interfaces and services in more detail, we first explain our message exchange patterns and how we map these to a Kafka-based messaging infrastructure that has been selected for the CONNECT project.

Message Exchange Patterns

We assume three different message exchange patterns to be used by the TAF and its communication partners and its environment:

Request/Response messaging is used for accessing services with a traditional communication pattern. In this case, the client needs to know the server identity, the service it provides, and how to address the server. The request/response pattern can also be used with session semantics (e.g., Trust Assessment Service). Here, the client first establishes a sessions with a special request/response before being able to send further, session-specific requests.

Publish/Subscribe is a message pattern for the exchange of notifications or events, usually between a known set of publishers and subscribers. In case subscribing entities are directly contacting publishers in order to signal their subscription, there is a direct coupling between both parties. This implies knowledge of the entities about the identities of the other entities. Explicit subscriptions are required when the consumer needs to tell the publisher the exact content it wants to subscribe to (i.e., publisher-based message filtering/dissemination) or when a subscription needs to managed explicitly in a service (e.g., specific timeouts for subscriptions).

However, when using a messaging middleware like Kafka, as we do in our prototype, publish/subscribe can also be implemented in a more decoupled way by directly using topics. So instead of knowing the list of subscribers (i.e., for publishers when sending notifications) or publishers (i.e., for subscribers when signaling subscriptions), it is now sufficient to only know the name of a topic to publish to or to subscribe to. This enables communication flows between entities – irrespective of mutual knowledge of their identities. Another caveat here is that Kafka topics allow new consumers to read older messages that have been published even before the subscription has occurred. This behavior needs to be addressed to prevent unwanted side-effects or unrealistic assumptions about the communication channel becoming an unbounded archive of all previous messages.

One-Way Notifications are messages sent from one entity to a potentially undefined group of receivers. Network scenarios for this type of messaging include broadcast, multicasts, or geocasts. When using Kafka, such broadcast-like notifications can either be implemented by using a well-known topics that all potential subscribers are expected to be consumers of (e.g., predefined broadcast topic). Alternatively, the sending entity must know all actual receivers and address them appropriately by using corresponding topics (e.g., cell-based/location-based topics for dynamic scenarios).

Table 4.1 illustrates which message exchange patterns are used as part of the interaction between the TAF and other entities.

Mapping to Kafka Messaging

In our prototype, Apache Kafka is used as messaging layer providing the communication abstraction between the TAF and TAF-external components.

Kafka is a messaging infrastructure that combines a publish/subscribe messaging system and a distributed commit log. It is thus a message queue system with an active broker that persists messages. Kafka decouples message producers and message consumers by introducing message topics that producers can write to and consumers can read from.

Due to the impact of this choice on the service and interface specifications, we first explain how we map inbound and outgoing communication between the TAF and other entities to Kafka-based messaging.

While Kafka generally provides a nice abstraction for low coupling of components, it also enforces a one-way, asynchronous message exchange style: (i) Producers append new messages to a topic whenever they want; (ii) Consumers read new messages from a topic whenever possible. The topic thus becomes an unbounded message queue that decouples the two communication partners. While Kafka is a natural fit for unidirectional messaging, one-way notifications (e.g., topic-scoped broadcasts) and for publish-subscribe based communication via topics, is it not good fit for request/response style synchronous communication.

For this reason, we specify a communication style to be used with the TAF whenever request/response communication is intended. This specification addresses (i) identification and addressing of sending and receiving entities, (ii) mapping of responses to corresponding requests, (iii) and message multiplexing/demultiplexing.

1. Any entity providing a service (a *server*) provides a dedicated Kafka topic to be used as conceptual inbox of that service at the entity. The name of the topic needs to be derived from the identity of the entity that provides the service and the name of the service. Requests to this service need to be sent to this topic. The entity providing the service is the only producer for that topic.

2. Messages that represent request messages that are sent to said inbox topic always need to contain the following mandatory fields of information:

   (a) An *identity of the entity sending* this request (i.e., the “client”). This allows the server to identify the client entity.

   (b) A *service type identifier* that specifies the service (to be specified for the service).

(c) A *message type identifier* that specifies the type of request within the service (to be specified by the service).

(d) A *topic* to be used by the server when sending back a *response* to the client. As clients can "anonymously" send messages to the client using the server's inbox topic, a server needs to know where to send the responses back to. Choosing an appropriate topic is at the discretion of the client. For instance, a client can provide its own inbox topic that it uses for all of it request/response communication.

(e) A *message identity* that uniquely identifies the request. This message identity is required to couple the subsequent response to the original request with transactional semantics. Hence, the client can later associate an incoming response to the original request by matching the identity of the response sender, the service type, and the message identifier.

(f) An *optional* blob of data representing the remainder of the *request message*.

3. A server that processes incoming requests based on the message format described above then generates a service-specific response. The response is again wrapped and sent to the client using the following additional fields:

   (a) An *identity of the entity sending* this response (i.e., the "server").

   (b) A *service type identifier* that specifies the service.

   (c) A *message type identifier* that specifies the type of request within the service.

   (d) A *message identifier* that is identical to the message identifier used in the original request.

   (e) An *optional* blob of data representing the remainder of the service-specific *response*.

4. In case of a client sending concurrent requests, the client needs to demultiplex different incoming responses in their receiving topic based on the tuple of (sender id, service type identifier, message identifier).

5. Another downside of the one-way messaging of Kafka is the loss of synchronicity in the client when issuing a request and then wanting to synchronously wait (i.e., blocking wait) for a corresponding response. As the response will now be handled by a different activity (i.e., Kafka topic consumer), language-specific mechanisms are required to define actions to be done once a response gets received (e.g., Futures, Callbacks, Continuations).

Please note that the specification of how to handle request/response communication might also apply to parts of the publish/subscribe message exchanges when subscription requests are handled explicitly. Here, the subscribing entity would invoke a request to the publishing entity to ask for the name of the topic that is used for notifications. If subscriptions are handled implicitly (i.e., by directly subscribing to a named, well-known Kafka topic), this does not apply.

Table 4.2: List of external interfaces with the involved entities, message exchange patterns and a high level description.

Service Name	Involved Entities	Message Exchange Pattern	Description

Connect DLT	DLT, TAF	Request/Response, Publish/Subscribe	TAF requests trust model template based on trust model template id; TAF subscribes for notifications of model template updates based on trust model template ids
Trust Assessment Service	TAF, Application/-Function	Request/Response, Publish/Subscribe	Applications initialize sessions with the TAF by stating releavant trust model template ids; In a session an application can invoke different kinds of TARs and/or subscribe to ATL updates.
V2X Message Listener	V2X Module, TAF	One way notification	Received CAMs / CPMs are forwarded to the TAF.
Node Discovery Interface	AND, ASD, TAF	Request/Response	The TAF requests information about surrounding nodes and/or MEC-services from the AND resp. ASD.
Trust Assessment Query Interface	Vehicle TAF, MEC TAF	Request/Response	The Vehicle TAF receives a trust value list containing trust opinions sent from the MEC.
Evidence Collection Interface	TAF, Evidence Source (e.g. Misbehavior Detection System, AIV)	Request/Response, Publish/Subscribe, One way notifications	The TAF either actively queries a trust source for evidence or receives updates about evidence from trust sources.

4.4.2 Trust Assessment Service

The Trust Assessment Service (TAS) represents the primary service a TAF provides to its clients — the continuous assessment of trustworthiness of relevant entities. In order to receive such a trust assessment, clients have to establish a session with the TAF and indicate the trust model template they are interested in. Once a session is established, the client can then either request trust assessments or subscribe for notifications in case of changes in the trustworthiness of the target entity.

This service represents a stateful, session-based protocol in which a client indicates interest in a target trust model (by specifying the trust model template to be used) and the TAF as the server ensures the processing of matching trust model instances for the lifetime of that session.

A client can establish zero or more sessions with a TAF. Each session requires exactly one trust model template to be used within this session. This means that a single client is allowed to run multiple sessions in parallel with the same TAF. Having multiple concurrent sessions is particularly necessary when a client is interested in trustworthiness assessments for different trust model templates at the same time.

Table 4.1: List of services and external interfaces and their usage in different standalone TAF deployment settings. Furthermore, the table lists the different role of the TAF in specific service settings.

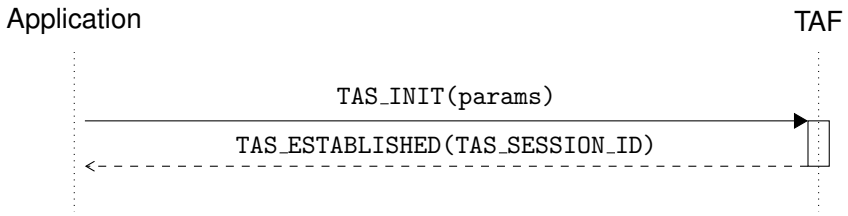
Service/External Interface	**TAF (Vehicle)**	**TAF (MEC)**
Trust Assessment Service	✓ S, Pub	*tbd.*
V2X Message Listener	✓ R	✓ R
CONNECT DLT	✓ C	✓ C
Trust Assessment Query Interface	✓ C/S	✓ C/S
Evidence Collection Interface	✓ C, R, Sub	✓ C, R, Sub
Node Discovery Interface	✓ C	✓ C

C: client role; **S**: server role; **R**: receiver/listener role; **Pub**: publisher role; **Sub**: subscriber role

For receiving ATLs during an established session, the client can choose between pull-based TARs (i.e., polling using request/response) or event-based subscriptions (i.e., publish/subscribe).

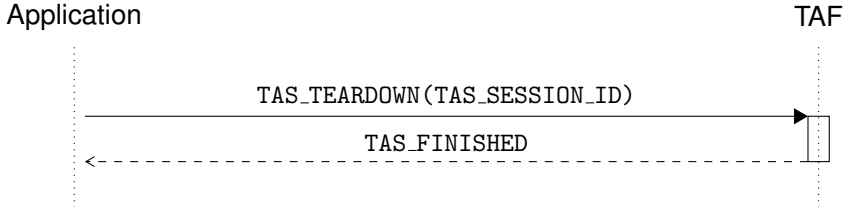
Session Initiation

In the session instantiation, the application tells the TAF which trust model template should be used by sending a TAS Initialization Message (TAS_INIT) which includes the Trust Model Template ID among other parameters. The TAF then internally prepares session handling for this session (e.g., by preparing internal data structures). Once ready, the TAF will respond with the ID of the new session.



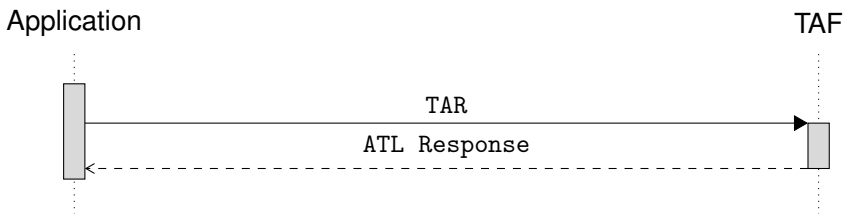
Session Tear-Down

Before shutting down, the application must signal the TAF to tear down the session. This allows the TAF to free all resources allocated for this session. For simplicity, the current design does not include a lease-based session model in which sessions would be purged automatically in case of client inactivity or missing renewals of the session.



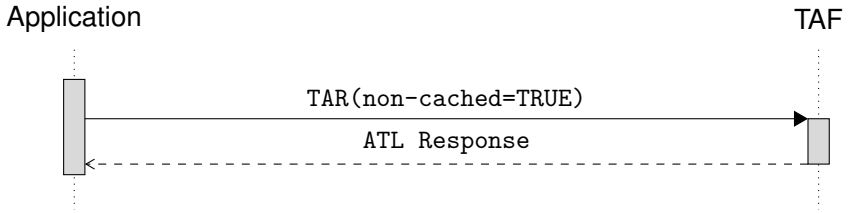
Regular Trust Assessment Requests

Once a session has been established, the application can send TARs to TAF whenever a recent trustworthiness assessment is needed. The TAF will respond to that request in an best-effort way and will potentially use cached results that had been calculated recently.



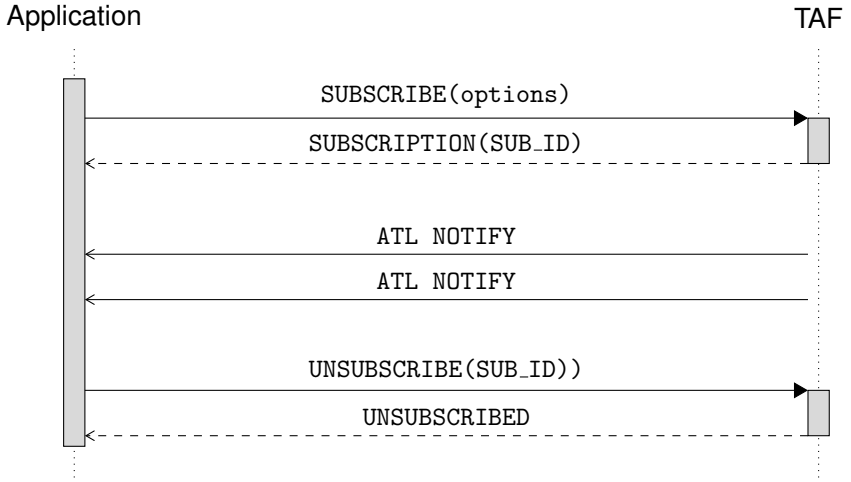
Non-Cached Trust Assessment Requests

In case an application requires freshly calculated assessments and does not accept results potentially precalculated recently by the TAF, it can request non-cached results from the TAF. This is a deliberate trade-off by the application to favor latest results over shorter response times.



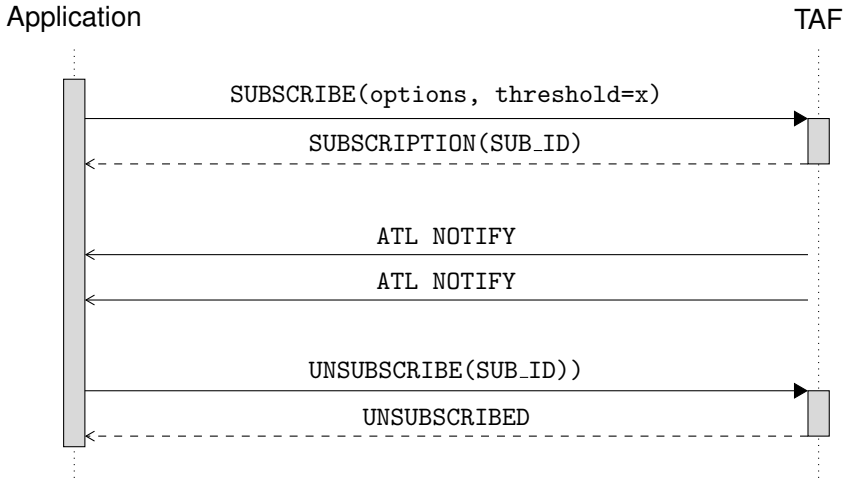
Subscription for Trust Level Changes

In addition to the previously described polling-based approaches, application can also subscribe for changes of ATLs. In the first approach, the application subscribes to any change. This means, whenever an ATL has been recalculated and is not identical to the previously published value, the application will get notified.



Subscription for Threshold-based Changes

An alternative approach for publish/subscribe include a threshold provided by the application during the subscription. In this case, the TAF will notify the application whenver an ATL has been (re-)calculated and the resulting value exceeds the threshold specified by the application.

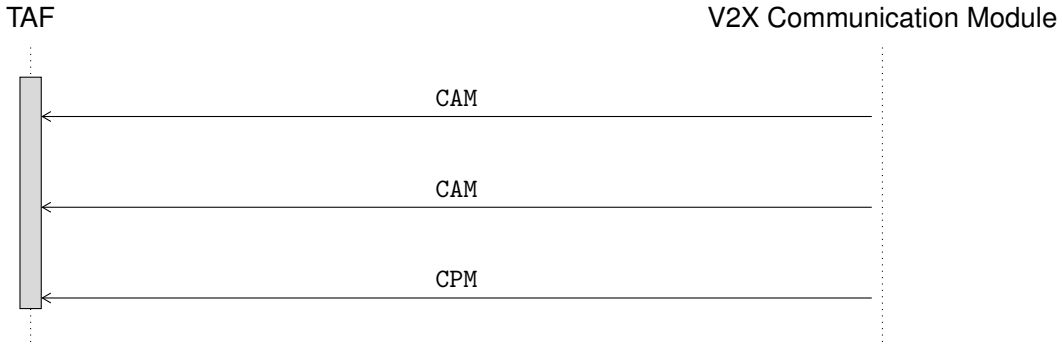


4.4.3 V2X Message Listener

The V2X Message Service is implemented by a TAF-external components that receives CAMs and CPMs (i.e., V2X Communication Module). This component provides a notification interface so that the TAF can register a listener and also receives notifications about incoming V2X messages.

V2X Message Notifications

By registering a listener, the TAF receives copies of incoming CAMs and CPMs. These messages can then be processed by TAF-internal components.



4.4.4 CONNECT DLT

The CONNECT DLT represents one of the main two repositories for trust model templates. Trust model templates that are directly installed next to an application or the TAF represent a first,

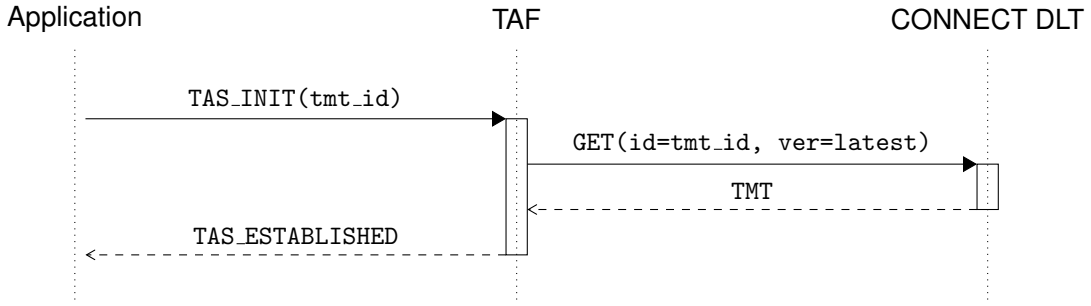
local source when trying to resolve trust model templates. The CONNECT DLT represents a second, remote source to retrieve templates. From the perspective of the TAF, the CONNECT DLT represents a distributed, read-only key-value store in which trust model templates are stored by ID and version number. So the key consists of a tuple `(trust model template identifier, trust model version)` and the value consists of a serialized represenation of the actual trust model template.

That said, we assume a dedicated query interface (e.g., HTTP-based web interface) in front of the actual distributed ledger so that the TAF does not have to interact with the ledger directly for performance reasons.

Furthermore, we assume two basic use cases in which the TAF will use this query interface: (i) retrieving a specific trust model template based on its ID and (ii) retrieving a trust model template based on its ID *and* referring to a specific version. In both cases, the TAF will have received a request to establish a trust assessment session that references a trust model templates not yet know by that TAF instance. Hence, the CONNECT DLT is used to resolve the template. In case the session request only specifies the trust model template identifier, we assume that the latest version of that template available in the DLT is to be used. If a specific version number for that template is used, the TAF will query for that specific version of the template. If the specified template (or version number) is unknown to the TAF, the remote resolution will fail and the TAF cannot instantiate the session and corresponding trust model instances.

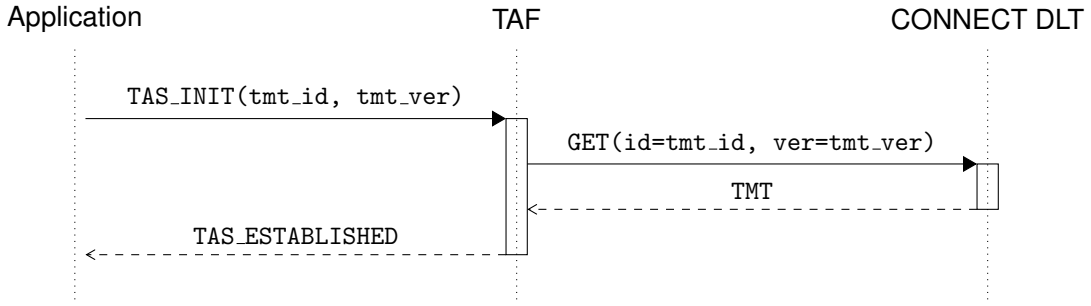
Remote Resolution of a Trust Model Template

When only using the trust model template ID, the TAF will try to resolve the latest version of that trust model template.



Remote Resolution of a Specific Version of a Trust Model Template

If the TAF needs to resolve a trust model template with a specific version number, this specific version is requested from the DLT.



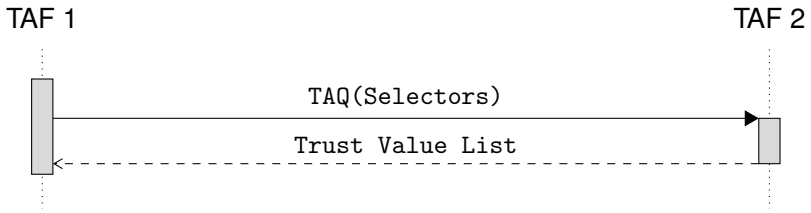
4.4.5 Trust Assessment Query Interface

The Trust Assessment Query Interface represents an auxiliary service of a TAF and enables other TAF instances to query trust assessments of that instance. Hence, this interface enables TAF-to-TAF interaction and can be seen as a first building block for federated behavior in which multiple TAFs enhance their own perspective by incorporating assessments provided by other TAFs.

Note that this interface is substantially different from the primary service interface of TAF, the trust assessment service. This service is used by applications or application functions that are collocated with the TAF. Furthermore, the trust assessment service determines which trust models are instantiated by TAF based on existing sessions. In turn, the Trust Assessment Query Interface represents a stateless, pull-based, read-only query protocol in which a TAF can ask for trust values of a remote TAF.

Trust Assessment Query

A trust assessment query (TAQ) contains a selector to identify trust values the querying TAF is interested in. A selector specifies a trust model template type and potential paths in corresponding trust model instances with optional attributes (i.e., to identify certain trust models and contained nodes). By using wildcards, a selector allows for broader queries, e.g., queries that can ask for any trust model instance in which a node with a certain ID is part of. The queried TAF then returns a potentially empty list of trust values satisfying the selector.



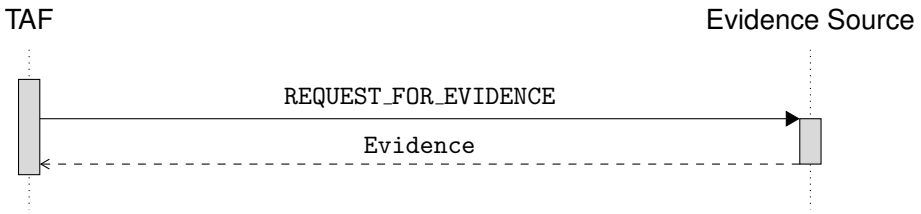
4.4.6 Evidence Collection Interface

The Evidence Collection Interface provides different mechanisms to collect information about other entities to be used by the trust source manager. This includes: (i) actively polling known nodes for evidence, (ii) subscribing for evidence notifications at know nodes, or (iii) receiving evidence updates via one-way messages.

The TAF implements the following interaction scheme so that the TSM can choose between them at runtime.

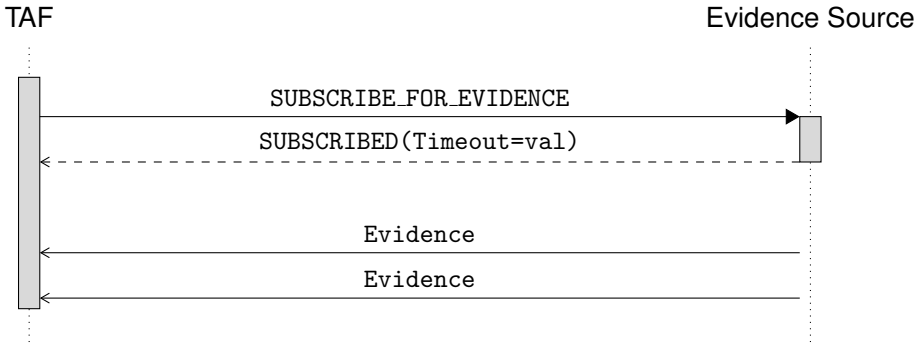
Pull-based Evidence Collection

In this case, the TAF knows about an available evidence source (e.g., by using the Node Discovery Interface) and specifically queries the target for evidence.



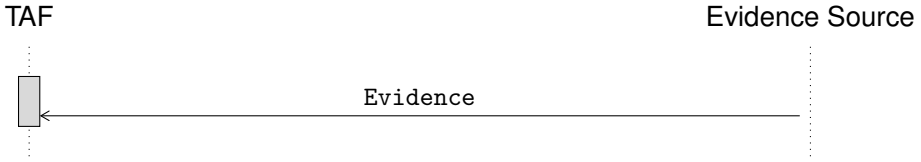
Subscription-based Evidence Collection

As an alternative to the previous interaction, a TAF can also use a known source of evidence and subscribe for evidence updates. These subscriptions have a timeout and need to be renewed.



One-Way Evidence Notifications

A third option is the reception of an unsolicited message that contains evidence. This evidence can the be used by the TAF, if applicable.



4.4.7 Node Discovery Interface

A Node Discovery Service allows the TAF to discover the presence of other entities (e.g., cars, MECs, TAF-enabled nodes). The TAF will use this service once it needs to query available communication partners (e.g., to check whether there is currently a MEC available) and it will only act as a client. We expect the service implementation to maintain lists of recently observed nodes and their associated information (e.g., for addressing).

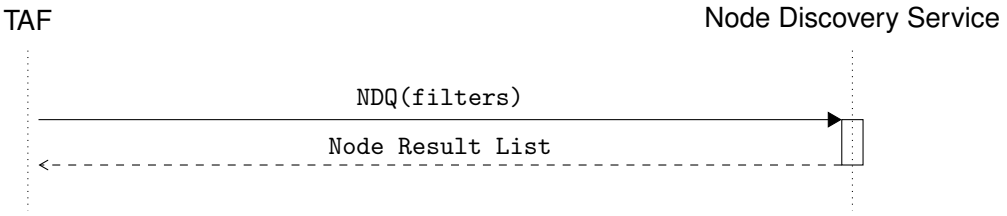
Node Discovery Query

A Node Discovery Query (NDQ) contains a potentially empty list of filters of discoverable nodes the TAF wants to query. Filters can include the identifier of a node or the type of the node (e.g.,

Table 4.3: List of internal and external interface exposure in the standalone TAF components.

Interface/Service	TAM	TMM	TSM	TLEE	TDE
External					
Trust Assessment Service	✓				
V2X Message Listener		✓	✓		
CONNECT DLT		✓	✓		
Trust Assessment Query Interface	✓				
Evidence Collection Interface			✓		
Node Discovery Interface	✓		✓		
Internal					
Trust Model Instance API	✓	✓	✓		
TLEE Invocation	✓			✓	
TDE Invocation	✓				✓

MEC). The Node Discovery Service returns a list of all matching nodes that the service has information on.



4.5 TAF Components & Internal Interfaces

This section describes the main components of the TAF and the internal interfaces by which they interact. These are not exposed to any component outside of the TAF and are described here mostly to document the architecture of the TAF prototype.

4.5.1 Trust Assessment Manager

The Trust Assessment Manager (TAM) is a central component for the TAF that is responsible for a number of tasks: (i) providing the trust assessment service functionalities to applications (including application session management); (ii) processing and incorporating changes of trust model instances when indicated by the trust model manager or the trust source manager; (iii) scheduling of TLEE computations; and (iv) scheduling of TDE function calls.

The TAM hereby decouples and compartmentalizes independent functions of the TAF:

1. It intentionally separates the observation of external changes from the update of the corresponding internal representations (i.e., trust model instances).

2. It intentionally separates the update of internal representations from the (re-)calculations of ATLs (i.e., TLEE) and trust decisions (i.e., TDE).

3. It intentionally separates the internal state handling from external requests (i.e., TAS).

In doing so, the TAM can deliberately schedule TLEE and TDE executions depending on (i) the number of pending change events for the trust model instance, (ii) the amount of trust model instance updates not yet reflected in the previous calculation, (iii) the staleness of previous calculation results stored in a cache, (iv) the demand for updated decisions based on actual sessions or pending requests (i.e., non-cached TARs), and (v) the current number of other concurrently issued executions for other trust model instances. Furthermore, the TAM can apply selective load-shedding in case of too many external updates. This gives us a high level of flexibility to configure and scale the TAF to different load scenarios and runtime environments with varying factors of parallelism, load-shedding, and caching.

Internal API: Trust Model Instance API

The TAF provides an internal, event-based API in which the trust model manager and the trust source manager components can emit events about updates in their observations. These changes are then processed by the TAM in order to update the trust model instances. This asynchronous API approach decouples the components and follows the single writer principle in a way that only the TAM has write access on all trust model instances. At the same time, this prevents write contention and concurrent writes between the trust model manager and the trust source manager. Instead, the trust assessment manager is expected to handle updates for the same trust model instance sequentially, although updates for different trust model instances can be executed in parallel.

4.5.2 Trust Model Manager

The Trust Model Manager is a component of the TAF responsible for managing Trust Model Templates and Trust Model Instances. As stated in Section 4.3, Trust Model Templates are application-specific templates created at design-time which specify all the relevant information for the instantiation of the Trust Model Instances. A Trust Model Template is always matched to a specific function/application and is meant to be used by the TAF to inform about which data the application needs as input to run a certain function, i.e., which data the TAF needs to output the Actual Trustworthiness Level for. Moreover, the Trust Model Template also specifies all the relevant Trust Objects, Trust Model Instantiation Policies, Trust Relationships, Trust Sources, and Trust Methods. Trust Model Instances are instantiated at run-time based on the Trust Model Templates and they are used to store Atomic Trust Opinions, Trust Opinions, and Actual Trustworthiness Levels assessed during run-time.

Trust Model Templates are stored in a Trust Model Template Database (TMT-DB) which is managed by the Trust Model Manager. Each Trust Model Template in the TMT-DB has a Trust Model Template ID (TMT-ID) associated with it, unique to each Trust Model Template. The TMT-ID values are pre-agreed upon and known to both the TAF and the application requesting a Trustworthiness Assessment, and, in most cases, do not change over time.

In the process of initializing a session with the TAF, an application will send the TMT-ID as part of the **initialization message**. The Trust Assessment Manager then forwards the TMT-ID to the Trust Model Manager which then either:

1. locates the corresponding Trust Model Template in the local TMT repository or, if not,
2. queries the Distributed Ledger Technology (DLT) where additional Trust Model Templates are for that specific TMT-ID.

If the specific TMT-ID is not found either in the local TMT-DB or in the DLT, then the Trust Model Manager returns a null value to the Trust Assessment Manager.

If, however, a corresponding Trust Model Template is found, then the Trust Model Manager will either:

1. create a Trust Model Instance based on this template and return a pointer to the instance to the Trust Assessment Manager, or
2. return a pointer to the Trust Model Template which has the corresponding TMT-ID to the Trust Assessment Manager.

The Trust Model Manager will only be able to instantiate *static* Trust Model Instances during initialization phase. This is because the structure of a static Trust Model Instance is based only on the fixed structure inside its Trust Model Template. The Trust Model Manager will not be able to instantiate *dynamic* Trust Model Instances during initialization phase because these types of instances are dependent also on the received CAMs and CPMs which the TAF has not yet started listening to during the initialization phase.

During the run-time phase, the TAF will receive a Trustworthiness Assessment Request (TAR) from an application, which will, among other pieces of information, contain a TMT-ID. This indicates that the application wants the TAF to perform trust assessment based on that specific Trust Model Template. The Trust Assessment Manager will use the TMT-ID to check whether there is already a Trust Model Instance associated with it. If there is, and if the Actual Trustworthiness levels have already been assessed and stored in the instance, the TAM will return the corresponding Actual Trustworthiness Levels to the application. If there is no Trust Model Instance for the corresponding TMT-ID yet, then the Trust Assessment Manager forwards the TMT-ID to the Trust Model Manager to start instantiating the instance(s).

In general, the Trust Model Manager is responsible for:

1. Creating appropriate Trust Model Instances during run-time based on the input it receives from the Trust Assessment Manager, as well as the incoming Cooperative Awareness Messages (CAMs) and Collective Perception Messages (CPMs).
2. Adding new trust objects to already existing Trust Model Instances based on appropriate input (CAMs and CPMs).
3. Deleting expired trust objects (such as CPMs representing vehicles which the TAF has not heard about in a while).
4. Deleting Trust Model Instances when a vehicle for which the instance was instantiated is no longer sending CAM or CPMs after a certain period of time.
5. Managing and updating the Trust Model Template Database.

4.5.3 Trust Source Manager

The **Trust Source Manager (TSM)** is a component of the TAF responsible for managing **Trust Source Quantifier Instances (TSQ-I)**.A TSQ-I is created at run-time based on the corresponding Trust Source Quantifier Template (TSQ-T) which is created at design time. Each TSQ-I is assigned to exactly one Trust Source. Trust Sources are external entities outside of the TAF. They provide evidence about the trustworthiness of a node or a data item. Examples for Trust Sources are a misbehavior detection system or the implemented security controls in a node. The TSQ-I receives evidence from the corresponding Trust Source and calculates an atomic trust opinion based on this evidence. To be able to do that, the corresponding TSQ-T describes how the evidence provided by the Trust Source should be interpreted. In addition, in the TSQ-T the corresponding formulas and calculation rules for calculating the belief, disbelief and uncertainty of the atomic trust opinion are defined. See Chapter 8 for the calculation rules and formulas of the trust sources misbehavior detection and implemented security controls.

There are two options for where the TAF obtains the TSQ-T. Both options are described in the following:

1. The TSQ-Ts could be stored in the TAF. They are created during design time and are installed together with the TAF on the vehicle/MEC when the corresponding node is set up by the manufacturer. Updates of the TSQ-Ts would be possible by an Over-The-Air update of the software of the TAF.

2. The TSQ-Ts could be stored in the cloud/DLT. If a TSQ-T is required for the calculation of an atomic trust opinion and this template is not stored locally in the TAF, the TAF requests the template from the cloud/DLT and will store it in its local memory.

In the trust model instance described in the previous section, it is specified which TSQ-T should be used to calculate the corresponding trust opinion. The TSQ-Ts have unique IDs (TSQ-T IDs). As already described in the previous section, an application sends the TMT-ID as part of the initialization message when a session is initialised between the application and the TAF. At this initialization phase, the TSM will process through the trust model instance and extract the corresponding TSQ-T IDs for each trust relationship. In addition to the TSQ-T IDs, further metadata is also provided in the trust model for each trust source. Based on this metadata a TSQ-I is created out of the TSQ-T. This metadata contains information for the calculation of the trust opinions. An example for this are weights describing the importance of each misbehavior detector when using a misbehavior detection system as a trust source. These weights are used as an input for the calculation rules and formulas described in the TSQ-T. In addition to the metadata necessary for the calculation rules and formulas, the trust model instance also includes information from where the TSM should request the evidence. For example, for misbehavior detection as a trust source, it is specified that the evidence will be requested from the misbehavior detection system running on the vehicle/MEC. For security controls as a trust source, it is specified that the evidence will be requested from the Attestation and Integrity Verification (AIV) component. Based on the TSQ-T ID and the corresponding metadata provided in the trust model instance, the TSM creates an instance based on TSQ-T, which is the TSQ-I.

As the TSM manages the TSQ-Is, it is aware of which instance needs evidence from which trust source. Therefore, the TSM handles the communication between the TAF and the trust sources. As already mentioned before, these are external entities that represent trust sources, such as a misbehavior detection system. The TSM either requests evidence or subscribes to the external

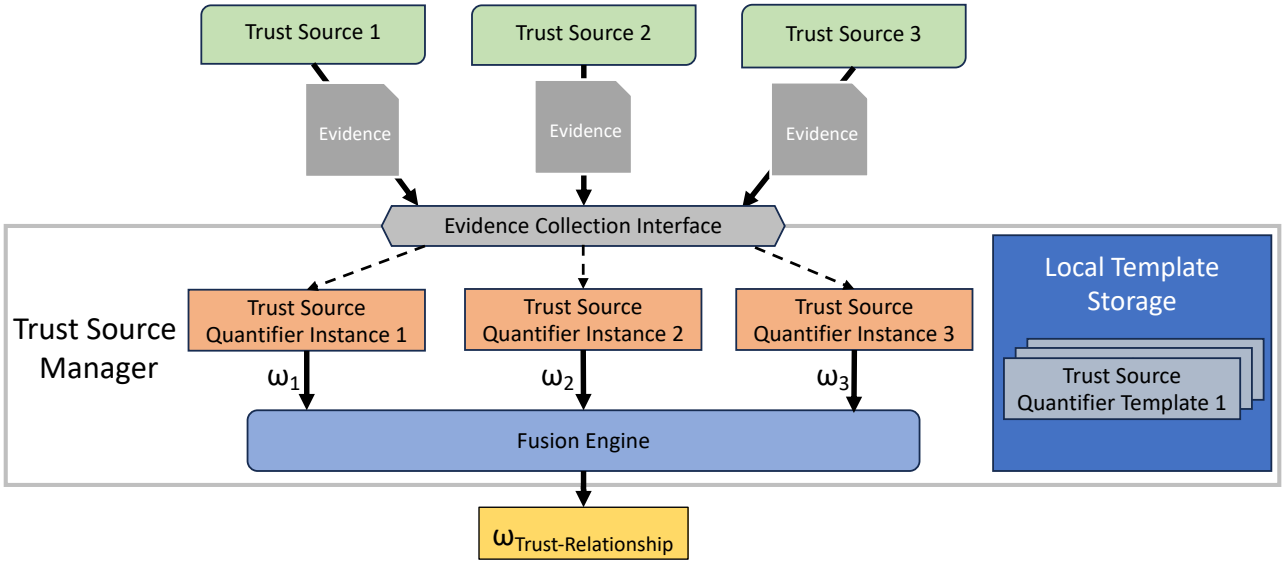


Figure 4.8: High level overview of the trust source manager handling receiving evidence and calculating a trust opinion for a single trust relationship.

entity so that it receives evidence as soon as the evidence has changed. The TSM also manages received evidence and forwards it to the corresponding TSQ-I. For this purpose, a unique identifier is necessary in each received evidence and which can be used by the TSM to map the evidence to the corresponding TSQ-I.

Based on the received evidence, each TSQ-I creates an atomic trust opinion. This is a subjective trust opinion calculated by a TSQ-I that has not been fused or discounted with trust opinions provided by other TSQ-Is. For one trust relationship in the trust model instance, several trust sources can be specified. The TSM is aware of which trust sources should be used for each trust relationship. After the corresponding TSQ-Is have calculated the atomic trust opinions for the specified trust sources of a trust relationship, the TSM fuses the atomic trust opinions to a trust opinion which is assigned to the corresponding trust relationship. For the fusion of the trust opinions, different fusion operators are possible. The operator to be used for the fusion is also specified in the trust model instance for each trust relationship. An overview of the described approach is shown in Figure 4.8.

4.5.4 Trustworthiness Level Expression Engine

As explained in the previous section, the TSM is responsible for creating, or quantifying Trust Opinions on a level of Trust Relationships (TO-TR). As defined in [10], the Trust Model combines various trust relationships among different trust objects. In response, TLEE is responsible for assessing the trustworthiness on the level of the trust model (i.e., the trust network), abbreviated as TO-TM. To calculate the trust opinion for the whole trust network, a set of functionalities are necessary that are provided by different modules in the TLEE. In the following section, we first explain the high-level architecture and the functional specifications of the different components of the TLEE. In the second half of the section, we describe the internal API of the TLEE with the TAM (as shown in figure 4.1), where we detail on the interface of the TLEE with the rest of the TAF components.

Architecture and functional specifications

In figure 4.9, we show the high-level TLEE architecture. The TLEE receives input from the TAM (I_TA_TLEE, through the `runTLEE()` function call) that triggers the execution of the four main components of the TLEE: *DSPG transformer*, *Expression Synthesizer*, *Meta-to-Concrete Expression Converter* and *Evaluator*. We explain `runTLEE()` function call in more detail later on. Besides the four main TLEE components, TLEE has an*Opinion Information Point* that caches the trust opinions for the trust relationships (TO-TRs), calculated by the TSM. Additionally, it is important to emphasize that the first three components (DSPG transformer, Expression Synthesizer and Meta-to-Concrete Expression Converter) operate symbolically by rewrite expressions on trust opinion variables; whereas, the last component calculates the final numerical trust opinion for the trust network.

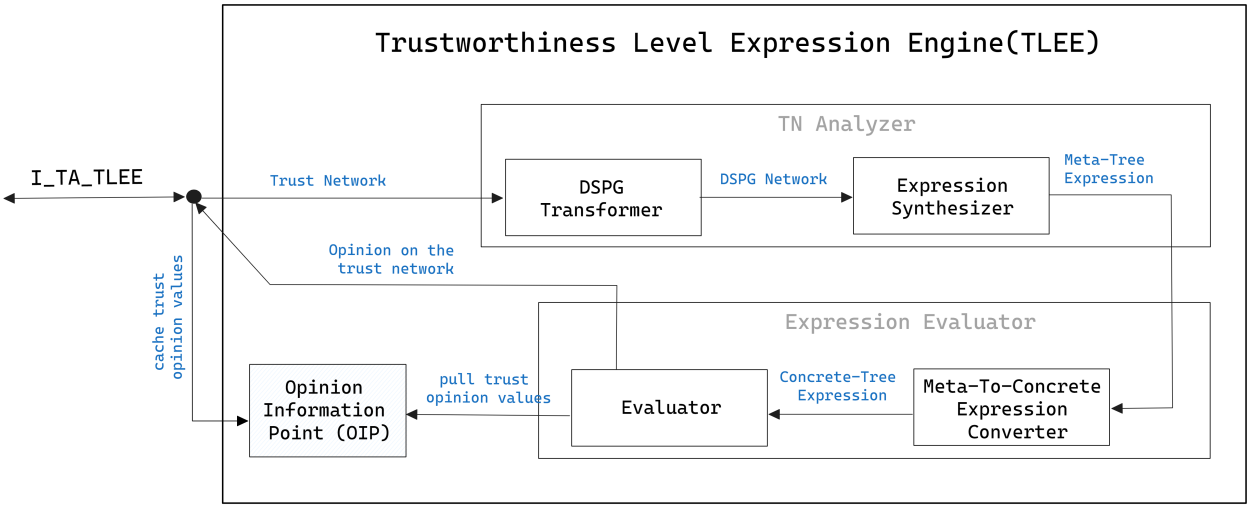


Figure 4.9: TLEE Architecture

As part of the TLEE, by employing trust discounting and fusion operations, the trust network (per proposition) can be effectively transformed into an equivalent single edge. This is shown in figures 4.10 and 4.11. Concretely, figure 4.10 shows an example of a trust model as received from the TMM, for four different propositions $Rdata$, G, $Cdata$ and $Ldata$. As it can be seen in 4.10, all the trust opinions that are depicted as part of this trust model are on a level of trust relationship (TO-TRs), and are therefore calculated by the TSM. Consequently, the TLEE is responsible for aggregating the trust opinions on the trust relationships (TO-TRs) that have been previously calculated by the TSM, based on which the final trust opinions on the level of trust network (per proposition) is derived (see figure 4.11). This consolidated edge encapsulates the comprehensive trust flow from the root of the trust model to the proposition (e.g.,, one of the leaves of the trust model), while considering the various trust relationships and their corresponding subjective opinions. The calculated opinions (ω^{VC}_{Rdata}, ω^{VC}_{G}, ω^{VC}_{Cdata} and ω^{VC}_{Ldata}) represent the actual trust levels (ATLs) calculated by the TLEE. Finally, we want to emphasize that we have already explained trust discounting and fusion as part of section 5.3.4. in [13], and we will provide the algebraic expressions of the operators later on in this section.

In the following, we explain in more detail the necessary processes to calculate the ATL values and the TLEE's functional specifications: the inputs, the outputs and a short summary of the functionality of the four main TLEE components.

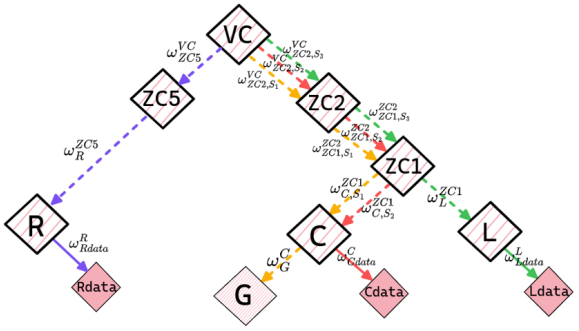


Figure 4.10: TM as received from the TMM

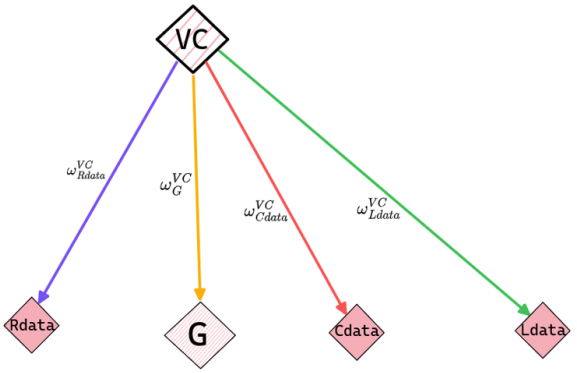


Figure 4.11: TM as aggregated by the TLEE

DSPG Transformer.

```
Input: trust network
Output: trust network in a DSPG format
```

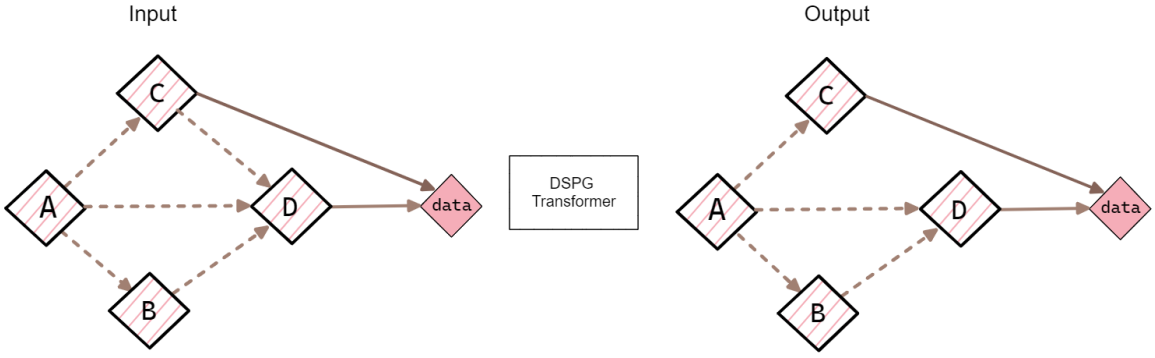


Figure 4.12: DSPG Transformer.

Figure 4.12 shows the input and the output of the DSPG transformer. DSPG stands for Directed Series-Parallel Graph (DSPG) [20], and we have already given a detailed explanation on Subjective Trust Networks and DPSG, as part of section 5.3.4. in [13]. To shortly recap: a subjective trust network (STN) is a graph representation of trust and belief relationships from agents, via other agents and sensors to target entities/variables, where each trust and belief relationship is expressed as a subjective opinion [20]. Please note that throughout our work we use the terms 1) trust model, 2) trust network and 3) STN, interchangeably. On the other side, for the trust network to be able to be analysed, i.e., for the operators for fusion and trust discounting to be applied to referral and functional trust relationships, trust network needs to be represented as a Directed Series-Parallel Graph (DSPG). DSPGs have a fundamental role in the TLEE implementation.

Definition 4.5.1 (Directed Series-Parallel Graph (DSPG) [20])**.** A graph is called a Directed Series-Parallel Graph (DSPG) if it can be decomposed as a combination of Series and Parallel graphs and it only consists of directed edges that form paths without loops from the source to the target.

By employing trust discounting and fusion operations, the DSPG trust network can be effectively transformed into an equivalent single edge, as previously shown in Figure 4.11. For that reason,

as part of the *DSPG Transformer*, we first check if the received trust network is in a DSPG format. If the trust network is a non-DSPG, then we do the necessary transformation.

To summarize, as input the DSPG transformer receives a trust network, which could be a complex non-DSPG or a DSPG graph. The functionality of this component is to first checks if the STN is a DSPG graph. If not, it synthesizes the complex non-DSPG to a DSPG graph. The transformation could be very simplistic, by removing edges of the graph that violate the DSPG properly. Alternatively, during the transformation, some additional information can be used as a heuristic (e.g., the opinions from the TSM) to make a more informed decision which edges to remove.

Expression Synthesizer.

```
Input: trust network in a DSPG format
Output: meta-tree expression
```

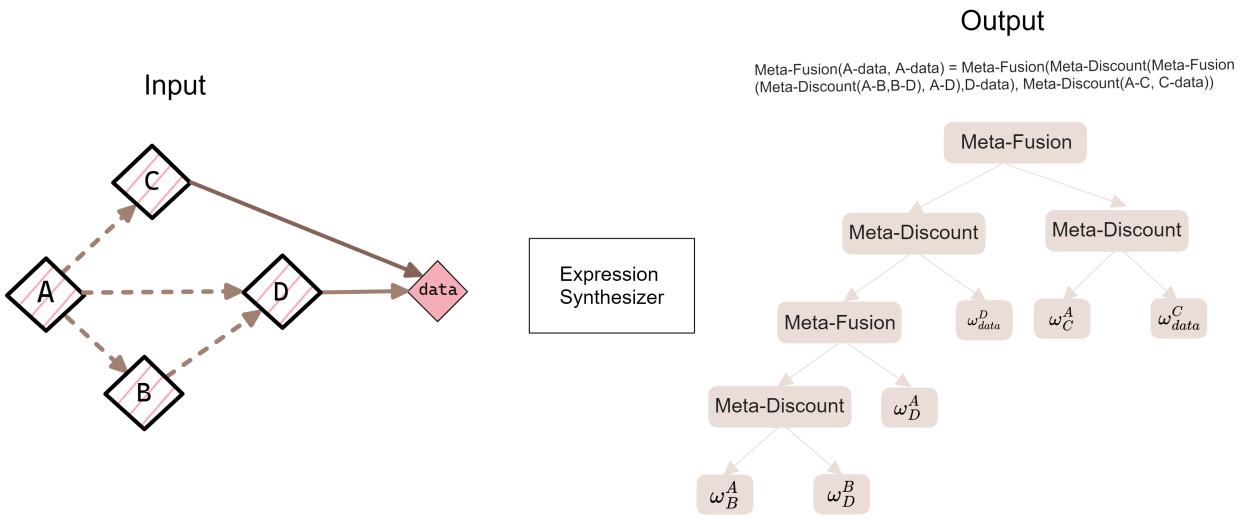


Figure 4.13: Expression synthesizer.

Figure 4.13 shows the input and the output of the *Expression Synthesizer*. As input this component receives a trust network in a DSPG format, and outputs a Meta-Tree Expression. The Meta-Tree Expression could be in tree or String format, as shown in the figure.

The functionality of this module is to build the Meta-Tree Expression from the trust network. This includes many sub-functionalities, where the two main one are: calculation of the nesting levels and graph analysis based on which the expression is built.

Step 1: Nesting levels calculation

In Figure 4.14, we show the calculated nesting levels for the exemplary DSPG. The concept of nesting level is important for analysing trust networks represented as a DSPG. To understand and define nesting levels, we first need to explain the concept of parallel-path subnetwork (PPS). First, a DSPG graph can consist of multiple subnetworks that themselves are DSPGs that can contain

parallel paths [20]. Therefore, we are interested in identifying subnetworks within a DSPG that contain parallel paths. A parallel-path subnetwork (PPS) in a DSPG is the set of multiple paths between a pair of connected nodes (e.g., nodes A and D, or A and $data$ in Figure 4.14). The nesting level of an edge reflects how many PPSs the edge is part of in the DSPG. Each edge has a specific nesting level greater or equal to 0. For example, a trust network consisting of a single trust path, has trust edges with nesting level 0, because the edges are not part of any subnetwork of parallel paths. If we refer again to Figure 4.14, then we can see that the edges between nodes A and D are part of two PPSs, and have nesting level of 2; whereas the edges between nodes A and C, C and $data$, and D and $data$ are part of one PPS, therefore have nesting level of 1.

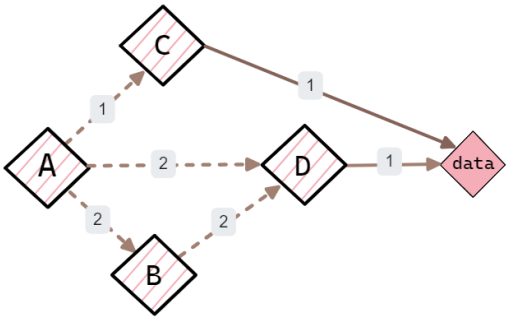
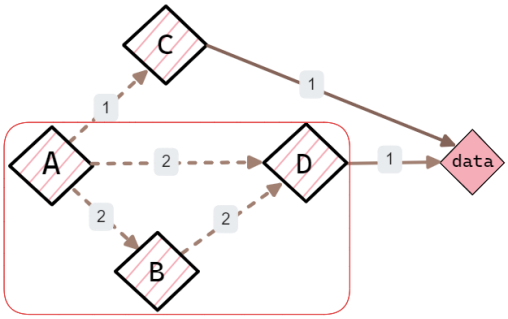


Figure 4.14: Nesting levels for the exemplary DSPG.

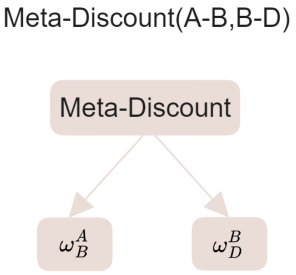
Step 2: Building Expression

Based on the calculated nesting levels, we start with building the expression. In the first phase in the process of building the expression, see Figure 4.15, we first discount the opinions ω_B^A and ω_D^B, to build the opinion ω_D^A. On the left hand-side of the figures you can see the trust network including its nesting levels, and how it changes during the graph reduction process. However, on the right hand-side of the figures, we show how the expressions are built iteratively in each of the phases. In the second phase, see figure 4.16, there are two opinions ω_D^A that need to be fused as part of this phase, to derive a single ω_D^A opinion. In the third and fourth phase, see figure 4.17 and figure 4.18, we discount the opinions ω_D^A and ω_{data}^D, and ω_C^A and ω_{data}^C, respectively. In this process we derive two opinions from A to $data$ (ω_{data}^A). In the final phase, see figure 4.19, we fuse both of the opinions from the fourth phase, and derive the final ω_{data}^A.

Our aim is to build an overall solution that is flexible and generalizable. For that reason, we have designed an engine that is math model agnostic and operates on a symbolic level, in order to support future extensions to other mathematical theories that enable dynamic trust assessment beyond SL. For instance, using other formal frameworks for trust assessment under uncertainty, e.g., Epistemic SL or EBSL (”evidence-based subjective logic), or even using homomorphic approaches to abstract from a particular mathematical model, as part of future research efforts. As part of our design of the TLEE, we have implemented this by first introducing meta-operators (meta-fusion and meta-discount), as part of the Expression Synthesizer module. Consequently, the output of this module is a meta-tree expression containing meta-operators that are instantiated to concrete operators in the next module.

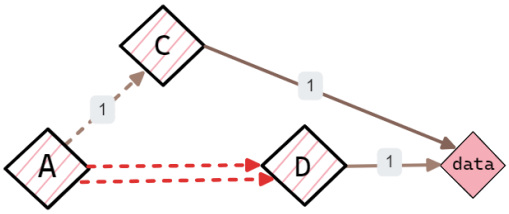


(a) Graph reduction

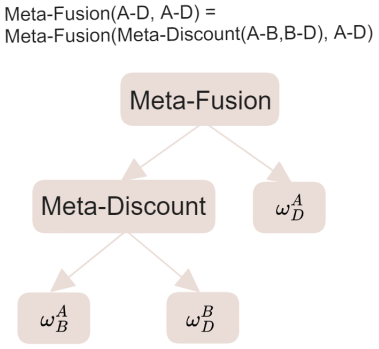


(b) Meta-tree expression

Figure 4.15: Building expression, phase 1.

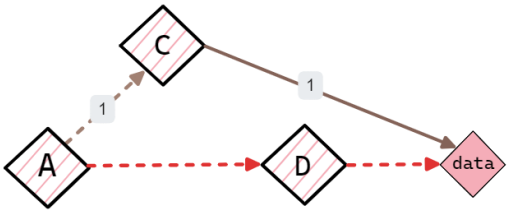


(a) Graph reduction

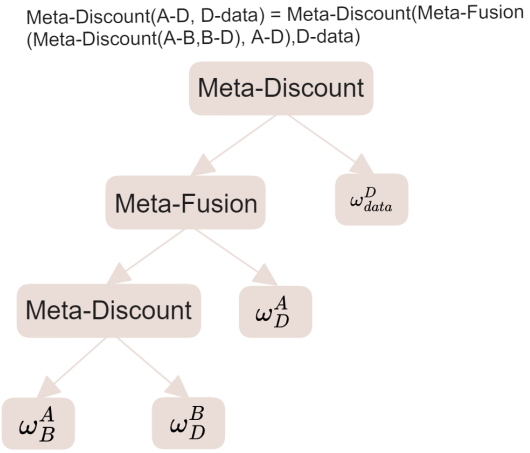


(b) Meta-tree expression

Figure 4.16: Building expression, phase 2.

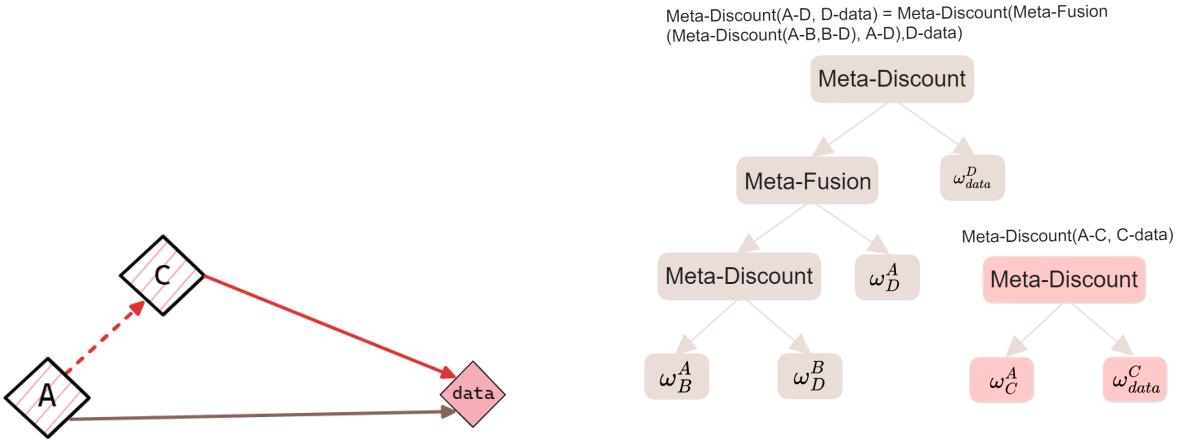


(a) Graph reduction



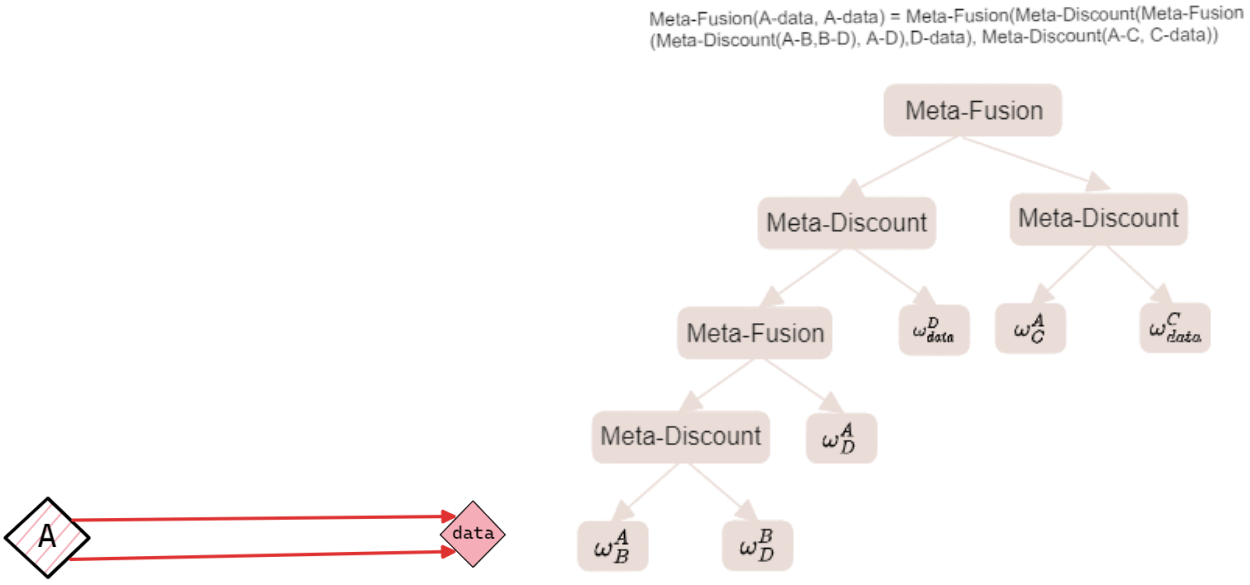
(b) Meta-tree expression

Figure 4.17: Building expression, phase 3.



(a) Graph reduction (b) Meta-tree expression

Figure 4.18: Building expression, phase 4.



(a) Graph reduction (b) Meta-tree expression

Figure 4.19: Building expression, phase 5.

Meta-to-Concrete Expression Converter.

```
Input: Meta-Tree Expression and (optionally) Math Model & Fusion Operators
Output: Concrete-Tree Expression (it could be in tree of String format)
```

We introduced a separate component *Meta-to-Concrete Expression Converter*, with a very simple functionality. Namely, the functionality of this component is to take 1) the meta-tree expression that is outputted in the previous component and 2) optionally a mathematical model (e.g., SL, EBSL, etc.) and return a concrete tree expression. In other words, the functionality of this component is to map the meta operators with concrete operators based on the specific mathematical model that can be given as an input parameter. Since as part of CONNECT project, we are exclusively building the TAF with the subjective logic theory, we have this parameter hard-coded in the

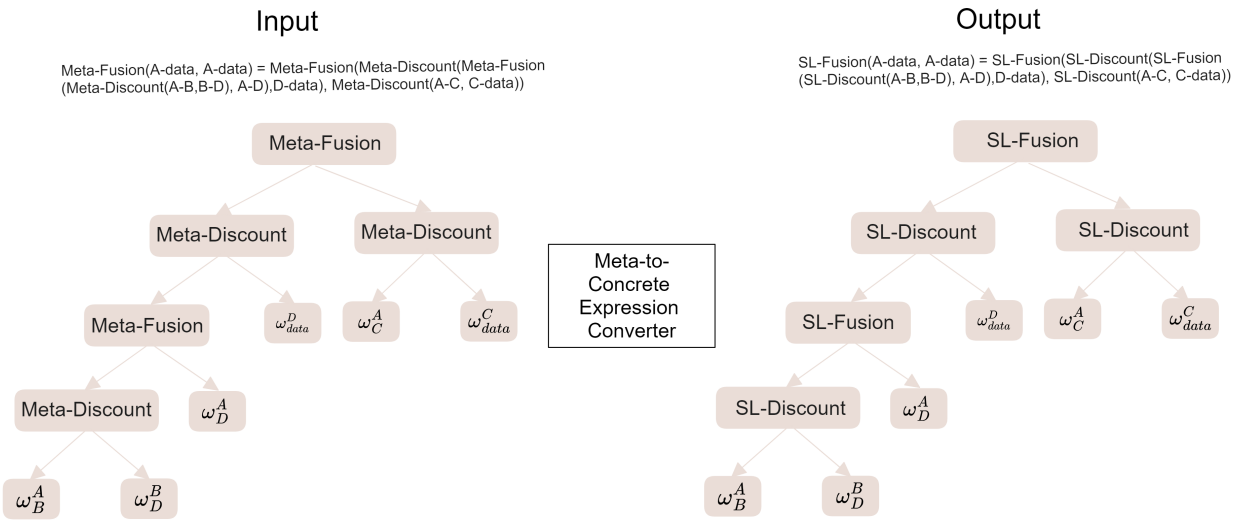


Figure 4.20: Meta-to-Concrete Expression Converter.

TLEE. Figure 4.20 shows the input and the output of the Meta-to-Concrete Expression Converter. As input the Meta-to-Concrete Expression Converter receives the Meta-Tree Expression from the previous module and the optionally the mathematical model according to which the operators will be instantiated. The Meta-to-Concrete Expression Converter outputs a Concrete-Tree Expression in tree or String format.

If the fusion operators are not already indicated in the trust model (i.e., the rewritten expression), then the component can also receive a parameter that specifies the concrete fusion operator (e.g., cumulative, average, etc.) that will be used for all the META-FUSION that do not have a specified fusion operator. Please note that here we assume that several mathematical models can implement the same type of fusion.

Evaluator.

```
Input: Concrete-Tree Expression + Trust Opinions on Trust Relationships
Output: Trust Opinion on Trust Network and (optional) stringified Expression
```

Figure 4.21 shows the input and the output of the Evaluator module. As input, the Evaluator receives the Concrete-Tree Expression from the previous module and the cached values of the Trust Opinions on Trust Relationships (TO-TR) that the Evaluator pulls from the Opinion Information Point (OIP) by demand. The TO-TRs are received from the TAM, calculated by the TSM and cached as part of the OIP component. To recap, as we already explained earlier in this section, until this component the TLEE engine rewrites the expressions symbolically, on a level of trust opinion *variables*. If we refer to figure 4.20, then those opinions variables are stored as part of the leaves of the tree. After 1) fetching the numerical values of the TO-TRs calculated by the TSM, and 2) applying the concrete discount and fusion operators, the final, numerical trust opinion for the trust network (TO-TM), or in other words, the ATL is calculated. The ATL(s) is the output of the Evaluator and, therefore, the TLEE. Alternatively, the stringified Concrete-Tree Expression can also be outputted. This stringified expression could be required in some applications and use cases where we need to track the exact expression that was used the calculate the final trust

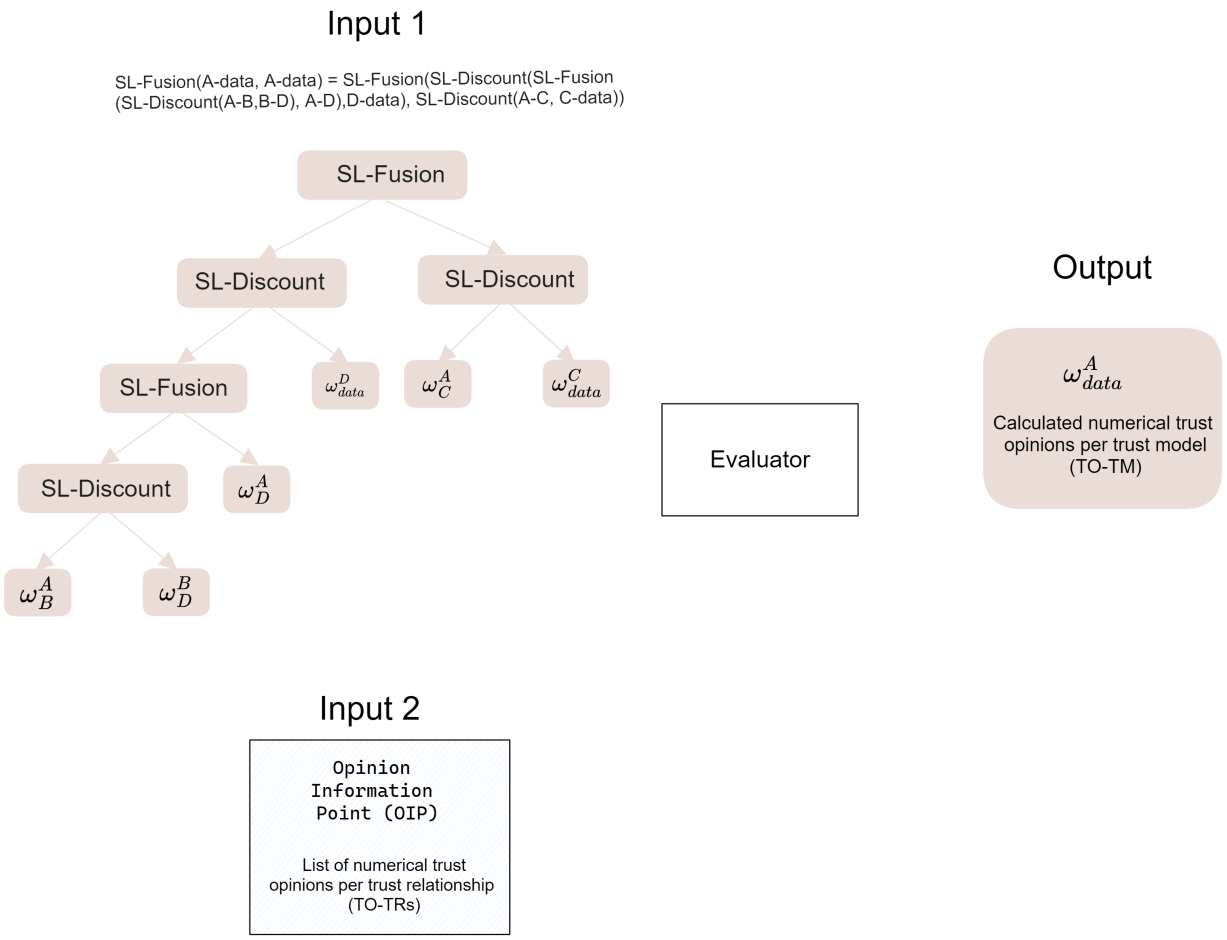


Figure 4.21: Evaluator.

opinions.

In the following subsection, we provide more details on the formalisms behind different trust fusion and trust discount operators.

Fusion and trust discount operators

As part of Subjective Logic theory, Jøsang has proposed five *fusion operators*. In the following, we explain each of the fusion operators, followed up with real-life, exemplary situations where the operators are applicable. Additionally, we provide the mathematical formulas for each of the operators. More details about the operators can be found in [20].

- **Belief Constraint Fusion** (BCF) [20] is suitable in situations where each agent expresses an opinion stating which variable values are the most correct, but in case of totally conflicting opinions the belief fusion is not allowed (i.e.,, an agreement between the expressed opinions is not possible). An example is when a group of friends try to agree on seeing a movie. If they share the same preferences for a movie, they can agree on watching that movie. However, if their preferences do not match then there is no agreement and they will not watch any movie together.

The algebraic expression for the Belief Constraint Fusion operator is given as follows [20]:

$$\omega_X^{A\&B}: \begin{cases} b_X^{(A\&B)}(x) = \frac{\mathrm{Har}(x)}{(1-\mathrm{Con})} \\ u_X^{(A\&B)} = \frac{u_X^A u_X^B}{(1-\mathrm{Con})}, \\ a^{(A\&B)}(x) = \frac{a_X^A(x)(1-u_X^A)+a_X^B(x)(1-u_X^B)}{(2-u_X^A-u_X^B)}, & \text{for } u_X^A + u_X^B < 2, \\ a^{(A\&B)}(x) = \frac{a_X^A(x)+a_X^B(x)}{2}, & \text{for } u_X^A = u_X^B = 1. \end{cases}$$

The term $Har(x)$ represents the relative harmony between constraints (in terms of overlapping belief mass) on x. The term Con represents the relative conflict between constraints (in terms of non-overlapping belief mass) between ω_X^A and ω_X^B.

$$Har(x) = b_X^A(x)u_X^B + b_X^B(x)u_X^A + \sum_{(x^A \cap x^B)=x} b_X^A(x^A)b_X^B(x^B),$$

$$Con = \sum_{(x^A \cap x^B)=0/} b_X^A(x^A)b_X^B(x^B).$$

- **Cumulative Belief Fusion** (CBF) [20] is when it is assumed that, as more and more (independent) sources are included, the amount of independent evidence increases. For example, an IT department could fuse the observed connections of a define network user over time, which produces an opinion with decreasing uncertainty about the most common locations of that user.

  Let ω^A and ω^B be source A and B's respective opinions over the same variable X. Let $\omega_X^{(A\diamond B)}$ be the opinion such that [20]:

  Case 1: For $u_X^A \neq 0 \vee u_X^B \neq 0$:

$$\begin{cases} b_X^{(A\diamond B)}(x) &= \frac{b_X^A(x)u_X^B + b_X^B(x)u_X^A}{u_X^A+u_X^B-u_X^A u_X^B} \\ u_X^{(A\diamond B)} &= \frac{u_X^A u_X^B}{u_X^A+u_X^B-u_X^A u_X^B} \\ a_X^{(A\diamond B)}(x) &= \frac{a_X^A(x)u_X^B + a_X^B(x)u_X^A - (a_X^A(x)+a_X^B(x))u_X^A u_X^B}{u_X^A+u_X^B-2u_X^A u_X^B} & \text{if } u_X^A \neq 1 \vee u_X^B \neq 1 \\ a_X^{(A\diamond B)}(x) &= \frac{a_X^A(x)+a_X^B(x)}{2} & \text{if } u_X^A = u_X^B = 1 \end{cases}$$

  Case 2: For $u_X^A = u_X^B = 0$:

$$\begin{cases} b_X^{A\diamond B}(x) &= \gamma_X^A b_X^A(x) + \gamma_X^B b_X^B(x) \\ u_X^{A\diamond B} &= 0 \\ a_X^{A\diamond B}(x) &= \gamma_X^A a_X^A(x) + \gamma_X^B a_X^B(x) \end{cases}$$

  where

$$\begin{cases} \gamma_X^A &= \lim_{\substack{u_X^A \to 0 \\ u_X^B \to 0}} \frac{u_X^B}{u_X^A+u_X^B} \\ \gamma_X^B &= \lim_{\substack{u_X^A \to 0 \\ u_X^B \to 0}} \frac{u_X^A}{u_X^A+u_X^B} \end{cases}$$

  Then $\omega_X^{(A\diamond B)}$ is called the cumulatively fused opinion of ω_X^A and ω_X^B, representing the combination of the independent opinions of sources A and B.

- **Averaging Belief Fusion** (ABF) [20] is when it is assumed that by including more sources does not imply that more evidence is supporting the final conclusion (i.e., dependence between sources is assumed). In ABF, a vacuous opinion contributes the same weight as every other opinion, therefore ABF has no neutral element and is idempotent. An example of this type of situation is when an examination committee tries to grade an student after having observed her dissertation.

  Let ω^A and ω^B be source A and B's respective opinions over the same variable X. Let $\omega_X^{(A\underline{\diamond}B)}$ be the opinion such that [20]:

  $Case\ 1 : u_X^A \neq 0 \vee u_X^B \neq 0 :$

$$\begin{cases} b_X^{(A\underline{\diamond}B)}(x) & = \frac{b_X^A(x)u_X^B + b_X^B(x)u_X^A}{u_X^A + u_X^B} \\ u_X^{(A\underline{\diamond}B)} & = \frac{2u_X^A u_X^B}{u_X^A + u_X^B} \\ a_X^{A\underline{\diamond}B}(x) & = \frac{a_X^A(x) + a_X^B(x)}{2} \end{cases}$$

  $Case\ 2 : u_X^A = u_X^B = 0 :$

$$\begin{cases} b_X^{(A\underline{\diamond}B)}(x) & = \gamma_X^A b_X^A(x) + \gamma_X^B b_X^B(x) \\ u_X^{(A\underline{\diamond}B)} & = 0 \\ a_X^{A\underline{\diamond}B}(x) & = \gamma_X^A a_X^A(x) + \gamma_X^B a_X^B(x) \end{cases}$$

  where

$$\begin{cases} \gamma_X^A & = \lim_{\substack{u_X^A \to 0 \\ u_X^B \to 0}} \frac{u_X^B}{u_X^A + u_X^B} \\ \gamma_X^B & = \lim_{\substack{u_X^A \to 0 \\ u_X^B \to 0}} \frac{u_X^A}{u_X^A + u_X^B} \end{cases}$$

  Then $\omega_X^{(A\underline{\diamond}B)}$ is called the averaged opinion of ω_X^A and ω_X^B, representing the combination of the dependent opinions of A and B.

- **Weighted Belief Fusion** (WBF) [20] is suitable in cases where the source opinions should be weighted as a function of the confidence of the opinions. When the input opinions have equally confident argument opinions, the fusion is averaging. In case one of the opinions is confident and the other is uncertain, then the confident opinion contributes the highest weight, however the combine confidence does not increase. WBF is commutative, considers a vacuous opinion as neutral element and is idempotent. An example of this type of situations is when, e.g. medical doctors express opinions about a set of possible diagnoses.

  Let ω^A and ω^B be source A and B's respective opinions over the same variable X. Let $\omega_X^{(A\widehat{\diamond}B)}$ be the opinion such that [20]:

  $Case\ 1 : (u_X^A \neq 0 \vee u_X^B \neq 0) \wedge (u_X^A \neq 1 \vee u_X^B \neq 1)$

$$\begin{cases} b_X^{(A\widehat{\diamond}B)}(x) = \frac{b_X^A(x)(1-u_X^A)u_X^B + b_X^B(x)(1-u_X^B)u_X^A}{u_X^A + u_X^B - 2u_X^A u_X^B} \\ u_X^{(A\widehat{\diamond}B)} = \frac{(2-u_X^A-u_X^B)u_X^A u_X^B}{u_X^A + u_X^B - 2u_X^A u_X^B} \\ a_X^{A\widehat{\diamond}B}(x) = \frac{a_X^A(x)(1-u_X^A) + a_X^B(x)(1-u_X^B)}{2-u_X^A-u_X^B} \end{cases}$$

$Case\ 2 : u_X^A = 0 \wedge u_X^B = 0$

$$\begin{cases} b_X^{(A\hat{\diamond}B)}(x) = \gamma_X^A b_X^A(x) + \gamma_X^B b_X^B(x) \\ u_X^{A\hat{\diamond}B} = 0 \\ a_X^{A\hat{\diamond}B}(x) = \gamma_X^A a_X^A(x) + \gamma_X^B a^B(x) \end{cases}$$

where

$$\begin{cases} \gamma_X^A = \lim_{u_X^A \to 0, u_X^B \to 0} \frac{u_X^B}{u_X^A + u_X^B} \\ \gamma_X^B = \lim_{u_X^A \to 0, u_X^B \to 0} \frac{u_X^A}{u_X^A + u_X^B} \end{cases}$$

$Case\ 3 : u_X^A = 1 \wedge u_X^B = 1$

$$\begin{cases} b_X^{(A\hat{\diamond}B)}(x) = 0 \\ u_X^{(A\hat{\diamond}B)} = 1 \\ a_X^{A\hat{\diamond}B}(x) = \frac{a_X^A(x) + a_X^B(x)}{2} \end{cases}$$

Then $\omega_X^{(A\hat{\diamond}B)}$ is called the weighted fusion opinion of ω_X^A and ω_X^B.

- **Consensus & Compromise Fusion** (CCF) [20] is suitable in cases where the input opinions are in conflict. CCF is designed to maintain shared beliefs from each source, and to transform conflicting beliefs into a compromise belief. If a consensus belief already exists, it is preserved, and compromise belief is formed when necessary. The resulting fused belief is uncertain in the presence of totally conflicting beliefs. CCF is idempotent, commutative and considers a vacuous opinion as neutral element. This is helpful in situations when different experts generate opinions identifying different options, such as when doctors with different expertise suggest potential causes in a diagnostic process. The fused opinion reflects the opinion of all experts, and illustrates the group as a whole is certain about a certain set of potential causes.

  The Consensus & Compromise Fusion opinion is defined as the result of the following three computation phases: (1) consensus phase, (2) compromise phase, (3) normalization phase.

  Step 1: Consensus step.

  The consensus step simply consists of determining shared belief mass between the two arguments, which is stored as the belief vector b_X^{cons}:

  $$b_X^{cons}(x) = min(b_X^A(x), b_X^B(x))$$

  The sum of consensus belief denoted b_X^{cons} is expressed as:

  $$b_X^{cons} = \sum_{x \in \mathcal{R}(X)} b_X^{cons}(X)$$

  The residue belief masses of the arguments are

  $$\begin{cases} b_X^{resA}(x) = b_X^A(x) - b_X^{cons}(x) \\ u_X^{resA} = b_X^B(x) - b_X^{cons}(x) \end{cases}$$

Step 2: Compromise step.

The compromise step redistributes conflicting residue belief mass to produce compromise belief mass, stored in b_X^{comp} X expressed by :

$$b_X^{comp} = b_X^{resA}(x)u_X^B + b_X^{resB}(x)u_X^A + \sum_{\{y \cap z = x\}} a_X(y/z)a_X(z/y)b_X^{resA}(y)b_X^{resB}(z) + \sum_{\{y \cup z = x\}\{y \cap z \neq \emptyset\}} (1 - a_X(y/z)a_X(z/y))b_X^{resA}(y)b_X^{resB}(z) + \sum_{\{y \cup z = x\}\{y \cap z = \emptyset\}} b_X^{resA}(y)b_X^{resB}(z)$$

Then the following quantities are computed:

Preliminary uncertainty mass:

$$u_X^{pre} = u_X^A u_X^B$$

Sum of compromise belief:

$$b_X^{comp} = \sum_{x \in \mathcal{P}(X) b_X^{comp}(x)}$$

In general, $b_X^{cons} + b_X^{comp} + u_X^{pre} < 1$, hence normalisation of b_X^{comp} is required:

Normalisation factor:

$$\eta = \frac{1 - b_X^{cons} - u_X^{pre}}{b_X^{comp}}$$

Because belief on X represents uncertainty mass, the fused uncertainty is

$$u_X^{A \diamond B}(x) = u_X^{pre} + \eta b_X^{comp}(X)$$

The compromise belief mass on X must then be set to zero, i.e. $b_X^{comp}(X) = 0$

Step 3: Merging consensus and compromise belief.

After normalisation, the resulting CC-fused belief is

$$b_X^{A \diamond B}(x) = b_X^{cons}(x) + \eta b_X^{comp}(x) \, \forall x \in \mathcal{R}(X)$$

The CC-fused opinion is then expressed as $\omega_X^{A \diamond B} = (b_X^{A \diamond B}, u_X^{A \diamond B}, a_X)$ This marks the end of the three-step process for consensus & compromise fusion.

The second operator that is fundamental to quantify trust is trust discount. Trust discounting innately is related to the concept of trust transitivity. We explained trust discounting in more detail as part of section 5.3.4. in [13].

As part of prior works, Jøsang has proposed three operators for trust transitivity/discount. We summarize those three operators in the following:

- **Uncertainty Favouring Trust Transitivity [21].** When agent A distrusts recommending agent B, it implies that A believes B doesn't know the truth on X, leading A to also not know the truth on X. This discounting operator is proven to be associative but not commutative, meaning the order of combining opinions matters. In paths with multiple recommending entities, opinion independence is assumed, prohibiting the same entity from appearing twice or more. Eq. 4.1 shows how to calculate the Uncertainty Favouring Trust discounted opinion.

$$\omega_X^{A:B} : \begin{cases} b_X^{A:B} & = b_B^A b_X^B \\ d_X^{A:B} & = b_B^A d_X^B \\ u_X^{A:B} & = 1 - (b_X^{A:B} + d_X^{A:B}) \\ a_X^{A:B} & = a_X^B \end{cases} \tag{4.1}$$

- **Base Rate Sensitive Trust Transitivity [21]** operator does not consider the base rate a_B^A when discounting, which may seem counter intuitive. This operator is equivalent to the trust discount operator from [20]. For example, in a scenario where a stranger seeks a car mechanic in a town known for honesty and asks the first person he meets to direct him to a good car mechanic. Since the stranger does not know this person he will have a fully uncertain trust opinion him ($b = 0, d = 0, u = 1$). However, the base rate will be high since it's known that citizens of this city are honest. Therefore this should impact somehow the discounted result. Eq. 4.2 shows how to calculate the Base Rate Sensitive Trust discounted opinion.

$$\omega_X^{A:B} : \begin{cases} b_X^{A:B} & = E(\omega_B^A) b_X^B \\ d_X^{A:B} & = E(\omega_B^A) d_X^B \\ u_X^{A:B} & = 1 - (b_X^{A:B} + d_X^{A:B}) \\ a_X^{A:B} & = a_X^B, \end{cases} \tag{4.2}$$

  where $E(\omega_B^A) = b_B^A + a_B^A u_B^A$.

  Nonetheless, this approach requires caution. For instance, if the stranger has high trust expectations but receives a recommendation from someone with uncertain beliefs, the resulting belief may seem counter intuitive. This issue could become more pronounced with longer trust paths. Therefore, a safety principle might be to apply base rate-sensitive discounting only to the last transitive link. In summary, the Uncertainty Favouring Trust discounting operator is safe and conservative, while the Base Rate-Sensitive operator can be more intuitive in some situations but requires careful application. Another solution to avoid this problem is to use only the uncertainty favouring operators and set the base rate to $\frac{1}{2}$ assuming that other information that might impact the base rate are already taken into account in the belief and disbelief.

- **Opposite Belief Favouring Trust Transitivity [21].** When agent A distrusts recommending agent B, it suggests A believes B consistently suggests the opposite of B's true opinion about the truth value of x. Consequently, A not only distrusts x to the extent that B suggests belief but also trusts x to the extent that B suggests disbelief because two disbeliefs combine to form belief in this scenario. This operator embodies the idea of "your enemy's enemy is your friend," applicable in certain situations. However, it should only be applied

when plausible.

$$\omega_C^{A:B} : \begin{cases} b_C^{A:B} & = b_B^A b_C^B + d_B^A d_C^B \\ d_C^{A:B} & = b_B^A d_C^B + d_B^A b_C^B \\ u_C^{A:B} & = 1 - (b_C^{A:B} + d_C^{A:B}) \\ a_C^{A:B} & = a_C^B \end{cases} \tag{4.3}$$

Note that using this operator can lead to an attack where an attacker who as a bad reputation to suggest the truth such that the agent will think that this truth is not the truth since the attacker is supposed to suggest the opposite of the truth.

As part of the future research efforts in CONNECT, we need to agree on the most adequate fusion and discount operators for our use cases, and even potentially develop our own operators based on our concrete requirements.

Internal API: TLEE Invocation

The TLEE is part of the TAF process and exposes a `runTLEE()` function that can be called by the TAM. The TAM will invoke the TLEE by calling the `runTLEE()` function whenever the TAM detects that a computation becomes necessary. The TAM will also invoke the `runTLEE()` function when a trust model instance has been initialized to allow the expression engine to prepare internal models and structures (i.e., the expressions). In this case, the function is called with `nil` as the value for the `values` parameter and is expected to return an empty list of ATLs.

`function runTLEE(trustmodelID, version, fingerprint, structure, values)`

A function that runs the computation of ATLs in a trust model instance.

`trustmodelID`	An identifier that uniquely identifies a trust model instance inside the TAF.
`version`	A numeric version number that identifies the version of the trust model instance. Both updates in the structure and changes in the values of the trust model instance will increase the version number. Higher version numbers represent more recent states of the trust model instance.
`fingerprint`	A numeric fingerprint of the structure of the trust model instance. This can be seen as hash of the graph topology of the trust model instance. If the fingerprints of two trust model instances are identical, they have the same topological structure. The exact numerical value of a fingerprint has no further semantics (cf. hashes) and should not be considered as increments over time.
`structure`	An internal representation of the structure of the trust model instance that technically represents an adjacency list. So it is a keyed list of trust objects IDs. For each trust object, the structure then contains a potentially empty list of the tuples (ID of target trust object, ID of trust relationship) that represents the outgoing edges from this trust object and the identifier for this relationship. Also, an optional fusion operator can be specified for each node.
`values`	A key/value list of trust relationship IDs and associated trust opinions to populate the structure with values.
returns	The function is expected to return a list of ATLs. This is represented by a key/-value list that contains the node IDs of all leaf nodes and their corresponding ATL values as calculated by the TLEE. If `nil` has been uses as `values` parameter, the function will return `nil` instead.

Furthermore, the following contract applies between the TAM (caller) and TLEE (callee):

1. The TAM will always call the TLEE for the first time with an empty list of values to allow for initialization.
2. The TAM will never call the TLEE twice with identical parameters. A subsequent call will at least require an update in the version (and corresponding changes).
3. If a subsequent function call contains the same trust model ID and the same fingerprint, the TLEE can be certain that the structure of that trust model instance has not changed.
4. The parameters passed from the TAM to the TLEE should be considered to be immutable. In particular, once passed, the TAM will never access or modify these values anymore, effectively transferring ownership to the TLEE.
5. The return values passed from the TLEE to the TAM should be considered to be immutable. In particular, once returned, the TLEE will never access or modify these values anymore, effectively transferring ownership to the TAM.
6. From the perspective of that TAM, the `runTLEE()` function is conceptionally a stateless and side-effect free function. The output of the function call is only determined by the

parameters passed to the TLEE in that function call. Hidden from the rest of the TAF, however, the internal implementation of the `runTLEE()` inside the TLEE can take advantage of internal states, caching, pre-computations etc. to optimize the function call execution.

4.5.5 Trust Decision Engine

The Trust Decision Engine (TDE) is a TAF component that provides a function to eventually decide on the trustworthiness of a node or data. The TAM will thus delegate trust decisions by calling the TDE whenever an eventual trust decision is required (e.g., after TLEE function calls or before sending a response to a TAR). It is left to the application sending the TAR to request a decision to be taken or just receive an ATL and take a decision by some internal logic. At this level, a trust decision is a binary decision that can take one of the following two forms:

1. POS_TD – implying that the Trust Decision is positive and that the entity whose trustworthiness was assessed in form of an ATL is **trustworthy** enough under current trustworthiness requirements.
2. NEG_TD – implying that the Trust Decision is negative and that the entity whose trustworthiness was assessed in form of an ATL is **not trustworthy** enough under current trustworthiness requirements..

A Trust Decision is made based upon the comparison of the ATL with the RTL. The TDE is invoked by the TAM when needed to operate upon the relevant trust model instance. The trust model instance contains both the ATL(s) and RTL(s) that need to be compared. In Deliverable 3.1, we have explained a method to compare the ATL and the RTL by calculating their projected probabilities and checking the condition $P(ATL) \geq P(RTL)$ and outputs the result as positive or negative trust decision. Whether this method is the most effective for making trust decisions is still to be investigated as part of the future work.

Internal API: TDE Invocation

By encapsulating this functionality into a separate component that provides a stateless function to make the actual trust decision, this implementation can be easily replaced by switching to a different TDE implementation.

```
function runTDE(values)
```

A function that executes the trust decision making.

`values`	A map with trust object identifiers as keys and tuples of corresponding ATLs and RTLs as values. So for each trust object to be decided upon, the data structure contains its ATL and RTL.
returns	The function is expected to return a map with the same number of entries as the input values. Each entry has the trust object identifier as key and the final trust decision for that object as the value.

Chapter 5

Federated TAF

This chapter describes a high-level architecture and major concepts for a federated TAF following the process outlined in Deliverable 3.1. The Deliverable 3.3 will then include a fine-grained architecture for the Federated TAF just like this document describes the fine-grained architecture for the Standalone TAF.

5.1 Motivations for the federated TAF architecture

In scenarios where trust evidence relevant to assessing a trust relationship is not generated close to the location of the TAF, challenges emerge, as it happens, for example, in the case of a vehicle aiming to offload to the MEC an application involving the processing of its sensor data. The application may ask the TAF at the MEC to assess the trustworthiness of the sensor, which can only be done by drawing upon trust evidence generated by the vehicle itself.

The most straightforward approach may be to transmit the raw trust evidence from the vehicle to the TAF at the MEC. This however presents several issues. Firstly, transferring trust evidence requires the availability of security measures and attestation capabilities to ensure integrity of the transmitted evidence, which may not be guaranteed for the channel supporting the sensor data transfer. Secondly, in real-time applications where latency is critical, transmitting the trust evidence may impose an extra burden, depending on the volume and frequency at which the evidence is generated. Finally, transmitting in-vehicle trust evidence raises significant privacy concerns as it could potentially aid in identifying the originating nodes: for example, the trust evidence may reveal details about the vehicle's on-board sensing setup or internal architecture.

To address these challenges and provide an integrated solution, CONNECT proposes a federated TAF architecture, to enable information exchange between TAFs located in distinct secure enclaves, whether on remote devices or on the same device. Federated TAFs exchange information in a compact form (an *opinion* in the subjective logic formalism, referred throughout the deliverable as *trust opinion*), rather than transmitting raw evidence. The semantics of the exchanged trust opinions are intelligible only by the federated TAFs, thus preserving the privacy of the involved entities. The interaction between the federated TAFs requires mutual trustworthiness assessment. This critical function is made possible by the CONNECT architecture, which provides the required attestation capabilities. Through the federated TAF architecture, leveraging

evidence held by another TAF becomes possible in a secure and privacy-preserving way.

The federation principle brings concrete benefits beyond facilitating the access to a remote trust source, seen in the offloading example. The federated TAF architecture can be ultimately leveraged to distribute the assessment of a network of trust relationships across two or several TAFs, with the overarching objective of exploiting all available trust evidence, independently of its location; and distributing the processing load, to ensure timely delivery of the requested trustworthiness levels even to time-critical applications.

5.2 Federated TAF functional specification

The standalone TAF (see Chapter 4) operates as an individual component (even if relying on trustworthiness evidence coming from outside the TAF itself). Its purpose is to instantiate a trust model and to dynamically assess the trustworthiness levels associated to the trust relationships in the model, based on the available trust evidence. The federated TAF is not a single component; rather, it is a system of multiple, interacting TAFs, operating with the objective of collaboratively assessing trust relationships. For comprehensibility and simplicity, this document will focus on pairwise interactions of two federated TAFs, but none of the concepts presented is limited to this.

The federated architecture refers to the capability of sharing information across different TAFs operating in distinct enclaves. The inner processing of each of the involved TAFs remains as specified in the standalone case. As the information within the TAF is formally represented using the subjective logic framework, trust opinions is the data type that is exchanged by the federated TAFs. **The functional specification of the federated TAF architecture summarizes in specifying protocols and interfaces for exchanging trust opinions between the involved TAFs**.

The federated TAF architecture involves two entities, TAF_A and TAF_B, deployed in separate enclaves. Both TAF_A and TAF_B have the capability of operating in standalone mode. Logically, the federation process takes place among a trust model instance living in TAF_A and a trust model instance living in TAF_B. For brevity and ease of reading, in the following we will simply say that the federation process takes place between TAF_A and TAF_B.

The federation process is initiated by a Federation Request (FR) from the **requesting** TAF_A, which queries the **petitioned** TAF_B for a specific trust opinion $\omega_X^Y\big|_{\mathrm{TAF}_B}$, referring to an identified property on a trust relationship between Y (trustor) and X (trustee), as evaluated by TAF_B. Through the federation process, TAF_A may subscribe to the dynamically evolving trust opinion $\omega_X^Y\big|_{\mathrm{TAF}_B}$, and receive continuous updates from TAF_B.

Notice that $\omega_X^Y\big|_{\mathrm{TAF}_B}$, which is the target of the federation process, could be an atomic opinion, available in TAF_B thanks to the access to relevant trust evidence; or any trust opinion which can be assessed by TAF_B, e.g., an Actual Trustworthiness Level (ATL).

To set up such a federation, TAF_A and TAF_B need to be able to uniquely identify the semantics of the trust opinion $\omega_X^B\big|_{\mathrm{TAF}_B}$. This can only happen if the target trust relationship and property belong both to TAF_A and TAF_B, and can be uniquely and unambiguously identified in both. This

requirement can be satisfied at design time, by appropriately designing the trust model templates for TAF_A and TAF_B and using consistent naming schemes, (e.g., URL-based). Details on the naming scheme will be presented in Deliverable 3.3.

In the following sections we further explore the architecture of the federated TAF with the help of some motivating examples.

5.3 Case 1 - Request for an atomic opinion relative to a trust source

In this section we describe the case where the federated TAF architecture is leveraged by the requesting TAF_A to gain access to the information provided by a remote source of trust evidence, aka trust source, accessible to the petitioned TAF_B.

Assumptions Let TAF_A be the requesting entity and TAF_B be the petitioned entity. We assume that a secure communication channel exists between the containers hosting TAF_A and TAF_B; furthermore, as per the CONNECT architecture, we assume that TAF_A and TAF_B have attestation capabilities, and therefore, can produce trust evidence that can be exchanged to mutually assure the trustworthiness of the federated TAFs. This includes evidence about the integrity of the communication channel and authenticity of the sender.

Notation Consider the simple trust model depicted in the right hand side of Figure 5.1, where $\omega^B_X\big|_{\mathrm{TAF}_B}$ denotes the trust opinion of node B (trustor) on node X (trustee), as evaluated in TAF_B. The two blocks on the side of the arrow between B and X represent two trust sources producing evidence relevant to the trust relationship. From these, the Trust Source Manager of TAF_B evaluates the atomic opinions $\omega^B_{X|T_1}\big|_{\mathrm{TAF}_B}$ and $\omega^B_{X|T_2}\big|_{\mathrm{TAF}_B}$. The trust opinion $\omega^B_X\big|_{\mathrm{TAF}_B}$ is evaluated by TAF_B by taking both atomic trust opinions into account. Notice that if only one atomic opinion $\omega^B_{X|T_1}\big|_{\mathrm{TAF}_B}$ is available, then $\omega^B_X\big|_{\mathrm{TAF}_B} = \omega^B_{X|T_1}\big|_{\mathrm{TAF}_B}$.

Federation Refer to Figure 5.1, where trust opinions and atomic opinions evaluated in the requesting TAF_A are depicted in red and trust opinions and atomic opinions evaluated in the petitioned TAF_B are depicted in blue. TAF_A needs to complete an assessment which requires to evaluate $\omega^A_X\big|_{\mathrm{TAF}_A}$, but since it does not have access to any relevant trust source, it cannot evaluate it directly. However, TAF_A knows that an atomic opinion $\omega^B_{X|T_1}\big|_{\mathrm{TAF}_B}$ is available in the TAF_B. The TAF_A model hence triggers a federation request addressed to TAF_B, requesting $\omega^B_{X|T_1}\big|_{\mathrm{TAF}_B}$. As visible in Figure 5.1, TAF_A does not know of the existence of the atomic opinion $\omega^B_{X|T_2}\big|_{\mathrm{TAF}_B}$ at TAF_B.

TAF_B responds with the required atomic opinion $\omega^B_{X|T_1}\big|_{\mathrm{TAF}_B}$, which TAF_A uses to evaluate $\omega^B_X\big|_{\mathrm{TAF}_A} = \omega^B_{X|T_1}\big|_{\mathrm{TAF}_B}$. Moreover, TAF_B provides its attestation evidence to TAF_A, so that TAF_A has the atomic opinion $\omega^A_{B|T_1}\big|_{\mathrm{TAF}_A}$. As a consequence, TAF_A can evaluate $\omega^A_B\big|_{\mathrm{TAF}_A} = \omega^A_{B|T_1}\big|_{\mathrm{TAF}_A}$.

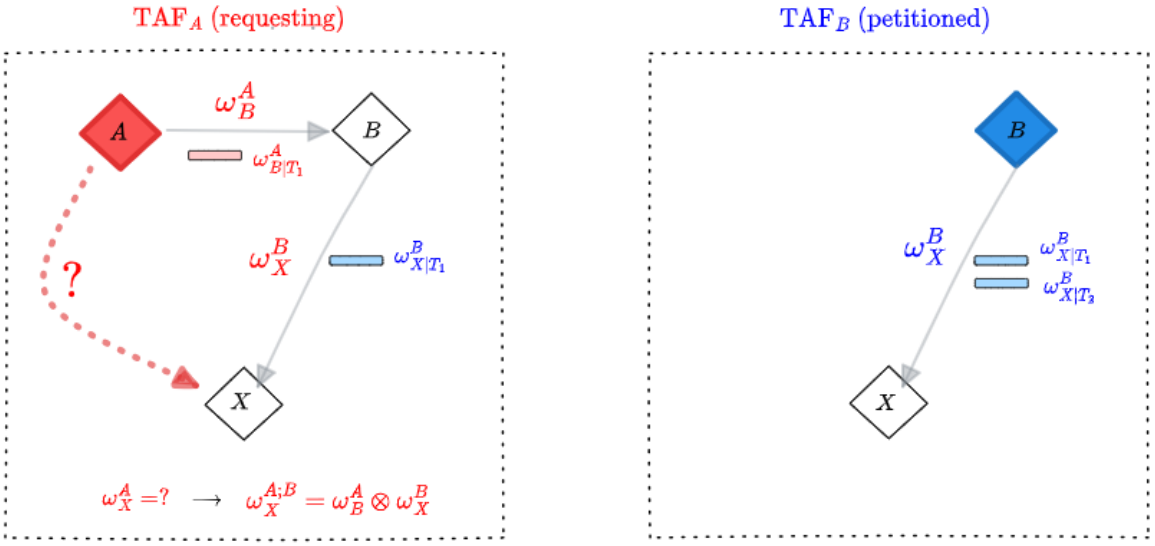


Figure 5.1: Case 1: Representation of the requesting TAF_A (left) and the petitioned TAF_B (right).

Finally, TAF_A can combine $\omega_X^B|_{\mathrm{TAF}_A}$ and $\omega_B^A|_{\mathrm{TAF}_A}$ to provide $\omega_X^A|_{\mathrm{TAF}_A} = \omega_X^{A;B}|_{\mathrm{TAF}_A}$ using trust discounting.

Once triggered, the federation is persistent in time. This means that whenever the value of the trust source $\omega_{X|T_1}^B|_{\mathrm{TAF}_B}$ changes because fresher evidence is observed at TAF_B, the new value is sent to TAF_A.

Design requirements For the above federation process to take place, the following requirements need to be fulfilled:

- The requesting TAF_A needs to know that the TAF_B can provide the atomic opinion $\omega_{X|T_1}^B|_{\mathrm{TAF}_B}$;
- The requesting TAF_A and the petitioned TAF_B reference the atomic opinion $\omega_{X|T_1}^B|_{\mathrm{TAF}_B}$ in a unique way.

The previous requirements are satisfied if the trust models in TAF_A and TAF_B share the notation and the semantics of the elements in Figure 5.1. This can be guaranteed by including the appropriate entities in the trust model template at design time.

Notice that the Trust Source Manager at TAF_A does not need to understand the semantics of the trust evidence producing the atomic opinion $\omega_{X|T_1}^B|_{\mathrm{TAF}_B}$. This constitutes an advantage towards the implementation of distributed trust assessment. Moreover, TAF_A does not need to know about the existence of the atomic opinion $\omega_{X|T_2}^B|_{\mathrm{TAF}_B}$ in TAF_B model. Notice that this has the consequence that $\omega_X^B|_{\mathrm{TAF}_A}$, evaluated by TAF_A thanks to a single trust source T_1, will have a different value than $\omega_X^B|_{\mathrm{TAF}_B}$, evaluated by TAF_B based on two trust sources T_1 and T_2. This is coherent with the subjective logic paradigm, where two trustors may have a different appraisal of the same trustee, if they have access to different trust evidence about it.

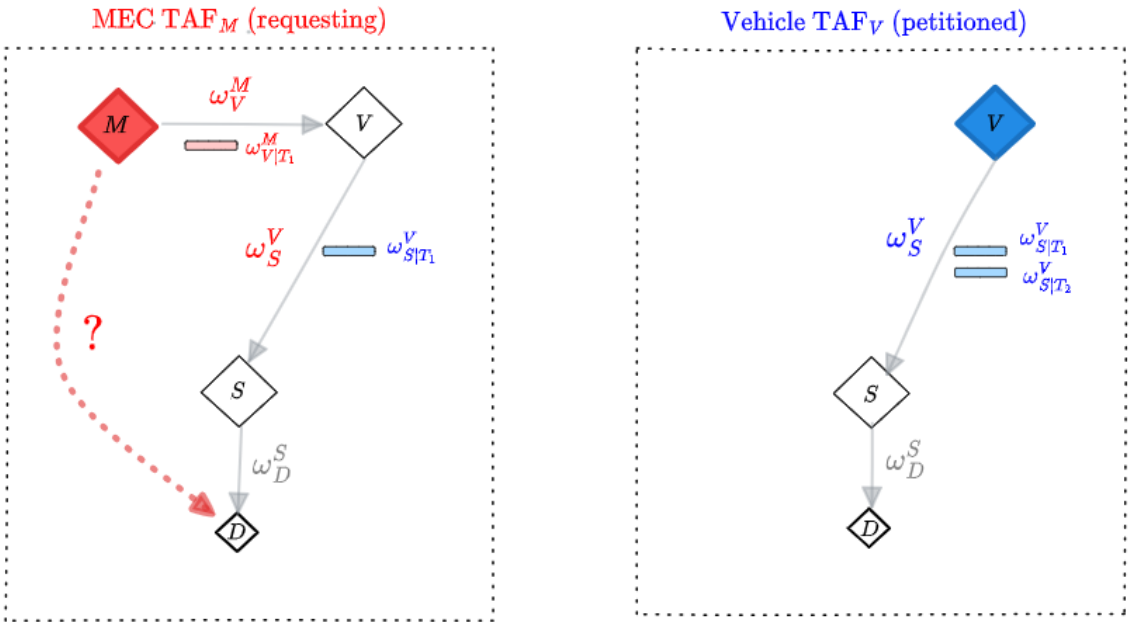


Figure 5.2: Case 1 Example: task offloading from the vehicle to the MEC.

Example To illustrate this case, we go back to the task offloading example. The TAF_M at the MEC and the TAF_V at the vehicle are depicted in Figure 5.2. During the normal operation time, the vehicle V runs an application which processes the datastream D from the sensor S. The application queries the TAF_V model to obtain the ATL $\omega_D^V\big|_{\mathrm{TAF}_V}$ about D. The TAF_V model has access to $\omega_D^S\big|_{\mathrm{TAF}_V}$, which is generated by S and is part of the datastream. Moreover, it has access to two sources of trust evidence $\omega_{S|T_1}^V\big|_{\mathrm{TAF}_V}$ and $\omega_{S|T_2}^V\big|_{\mathrm{TAF}_V}$, which are relevant for $\omega_S^V\big|_{\mathrm{TAF}_V}$. It can then produce the required ATL $\omega_D^V\big|_{\mathrm{TAF}_B}$ as $\omega_D^{V;S}\big|_{\mathrm{TAF}_V}$.

When the offloading operation begins, the processing task is instantiated at the MEC. The MEC receives a datastream containing D and ω_D^S. The application now asks the TAF_M the ATL $\omega_D^M\big|_{\mathrm{TAF}_M}$.

The TAF_M does not have access to a trust source on S, but knows that V has access to the relevant atomic opinion $\omega_{S|T_1}^V\big|_{\mathrm{TAF}_V}$, and initiates the federation process. Notice that the atomic opinion $\omega_{S|T_2}^V\big|_{\mathrm{TAF}_V}$ is relevant when the application is running in V, but it is not when the datastream is offloaded, hence it does not appear in TAF_M. This may be because in the two cases the data traverse different paths in the vehicle's onboard architecture.

The federation instance triggers the transmission of the trust evidence from TAF_V to TAF_M. So finally, the TAF_M model is able to assess the required ATL as $\omega_D^{M;V;S}\big|_{\mathrm{TAF}_A}$.

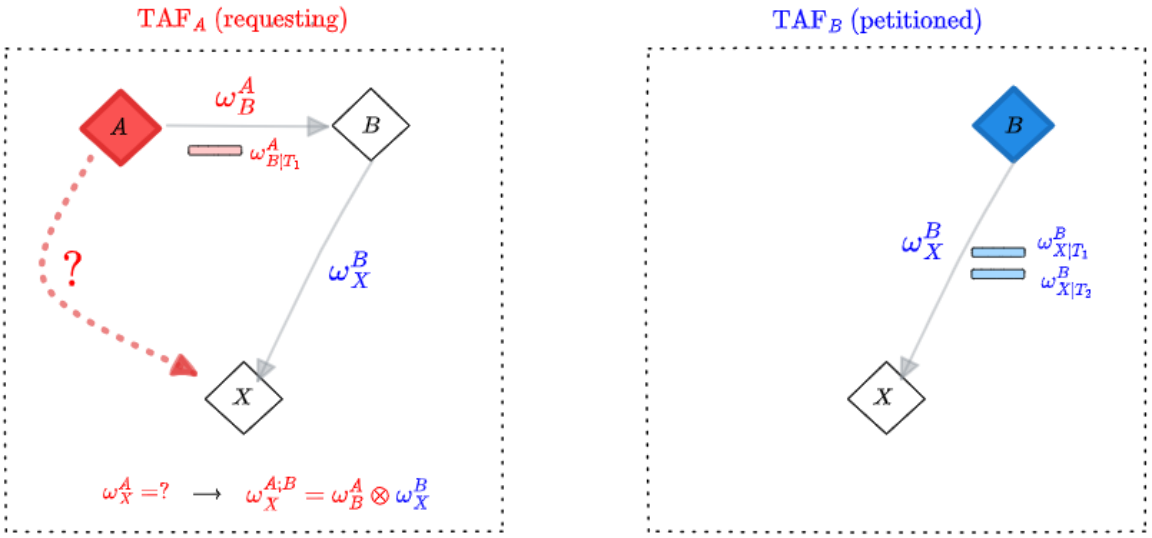


Figure 5.3: Case 2: Representation of the requesting TAF_A (left) and of the petitioned TAF_B (right).

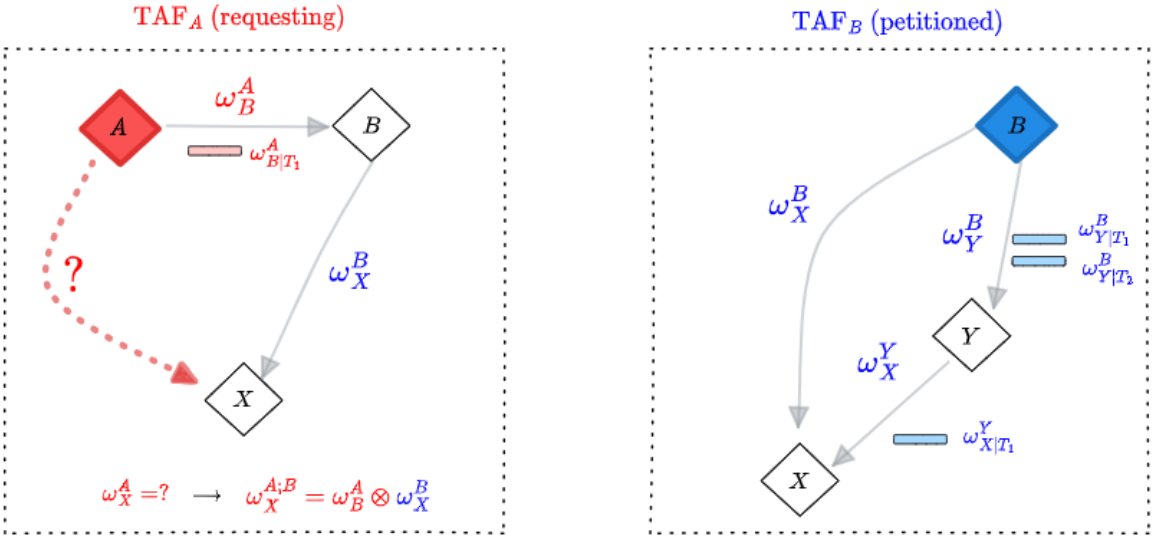


Figure 5.4: Case 2: Representation of the requesting TAF_A (left) and of the petitioned TAF_B (right).

5.4 Case 2 - Request for a trust opinion for a single trust relationship

In this section we consider the case where the federated TAF architecture is leveraged by the requesting TAF_A not simply for querying an atomic opinion relative to an unaccessible trust source, but to require the result of a trust opinioncomputation performed by the petiotioned TAF_B. This case is very similar to case 1, just the way how the opinion is created differs.

Assumptions We consider the same assumptions as in the previous case. As before, TAF_A is the requesting entity TAF_B is the petitioned entity.

Federation The TAF_A and TAF_B are depicted in Figure 5.3. As in Case 1, the TAF_A model needs to assess the opinion $\omega_X^A\big|_{\mathrm{TAF}_A}$, and it cannot evaluate it directly since it has not access to any relevant trust source. This time, TAF_A knows that TAF_B is able to assess the trust opinion $\omega_X^B\big|_{\mathrm{TAF}_B}$. Then TAF_A initiates the federation process,requesting $\omega_X^B\big|_{\mathrm{TAF}_B}$.

TAF_B receives the federation request. Notice that, by the TAF_B perspective, this is in principle equivalent to receiving a TAR for $\omega_X^B\big|_{\mathrm{TAF}_B}$ by an application. TAF_B responds with the required $\omega_X^B\big|_{\mathrm{TAF}_B}$, which it evaluates as fusion of the trust sources $\omega_{X|T_1}^B\big|_{\mathrm{TAF}_B}$ and $\omega_{X|T_2}^B\big|_{\mathrm{TAF}_B}$. Furthermore, TAF_B sends its attestation evidence, which constitute a trust source for TAF_A, which evaluates the atomic opinion $\omega_{B|T_1}^A\big|_{\mathrm{TAF}_A}$.

Finally, TAF_A is able to discount the received opinion $\omega_X^B\big|_{\mathrm{TAF}_B}$ by $\omega_B^A\big|_{\mathrm{TAF}_A}$, and to evaluate the required $\omega_X^A\big|_{\mathrm{TAF}_A} = \omega_X^{A;B}\big|_{\mathrm{TAF}_A}$.

Federation (variation of TAF_B) To better illustrate this case we consider the federation process in the same setting, with the variation of the TAF_B depicted in Figure 5.4. As before TAF_A is initiating the federation to require $\omega_X^B\big|_{\mathrm{TAF}_B}$.This time TAF_B evaluates $\omega_X^B\big|_{\mathrm{TAF}_B}$=$\omega_X^{B;Y}\big|_{\mathrm{TAF}_B}$, as it would do in receiving a TAR for $\omega_X^B\big|_{\mathrm{TAF}_B}$.

Trust model design requirements For the above process to take place the following requirements need to be fulfilled:

- The requesting TAF_A needs to know that the petitioned TAF_B is able to evaluate $\omega_X^B\big|_{\mathrm{TAF}_B}$;
- The requesting TAF_A and the petitioned TAF_B reference trust opinion $\omega_X^B\big|_{\mathrm{TAF}_B}$ in a unique way.

The previous requirements are satisfied if TAF_A and TAF_B share the notation and the semantics of the elements in Figure 5.3. Notice that the requesting TAF_A explicitly encodes the fact that $\omega_X^B\big|_{\mathrm{TAF}_B}$ is evaluated by TAF_B. Notice that this implies that TAF_A does not need to know how $\omega_X^B\big|_{\mathrm{TAF}_B}$ is evaluated by TAF_B, which is an advantage towards computation distribution across TAFs.

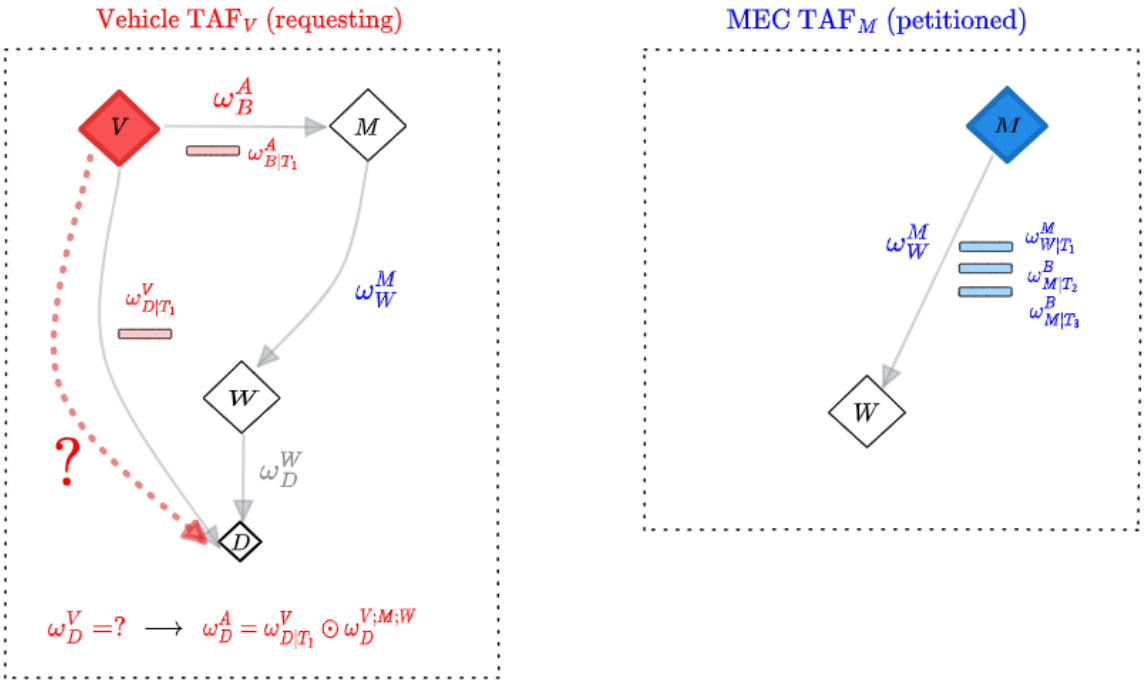


Figure 5.5: Case 2 Example: MEC-based V2X Node Trustworthiness Assessment Service

Example We consider the example of the MEC-based V2X Node Trustworthiness Assessment Service, which is a CONNECT use case scenario involving a vehicle and the MEC. The vehicle TAF_V and the MEC TAF_M are depicted in Figure 5.5.

A vehicle V receives V2X data D from another vehicle W, and issues a TAR to TAF_V for the ATL $\omega_D^V|_{\mathrm{TAF}_V}$. The vehicle's onboard system provides the misbehaviour detection service, whose output is used by the TAF_V as source of evidence, making the atomic opinion $\omega_{D|T_1}^V|_{\mathrm{TAF}_V}$ available in TAF_V. Moreover, TAF_V knows that it can queryTAF_M for $\omega_W^M|_{\mathrm{TAF}_M}$, and hence initiates a federation process. Receiving $\omega_W^M|_{\mathrm{TAF}_M}$ will enable TAF_V to use this information in the evaluation of the ATL $\omega_D^V|_{\mathrm{TAF}_V}$, thus leveraging trust evidence that is not directly accessible to TAF_V.

The TAF_M model is dynamically evolving, as TAF_V model. TAF_M is updated to contain the trust object W in the following cases: either the vehicle W uploads its vehicular trustworthiness claims to the MEC; or the vehicle W has been the target of a misbehaviour report sent by another vehicle to the MEC. Notice that both vehicular trustworthiness claims and misbehaviour reports are treated by the TAF_M as trust evidence, and generate atomic opinions $\omega_{W|T_i}^M|_{\mathrm{TAF}_M}$.

When the TAF_M receives the federation request, two cases are possible. In the first case, TAF_M contains the trust object W, and is able to positively respond to the federation request, by evaluating $\omega_W^M|_{\mathrm{TAF}_M}$ using the available trust sources. In the second case, W is not in the TAF_M model, and the federation request is refused. TAF_V may try and successfully initiate the federation instance again in the future, because the dynamic nature of the TAF_M model will allow to create the W trust object, as soon as trust evidence becomes available.

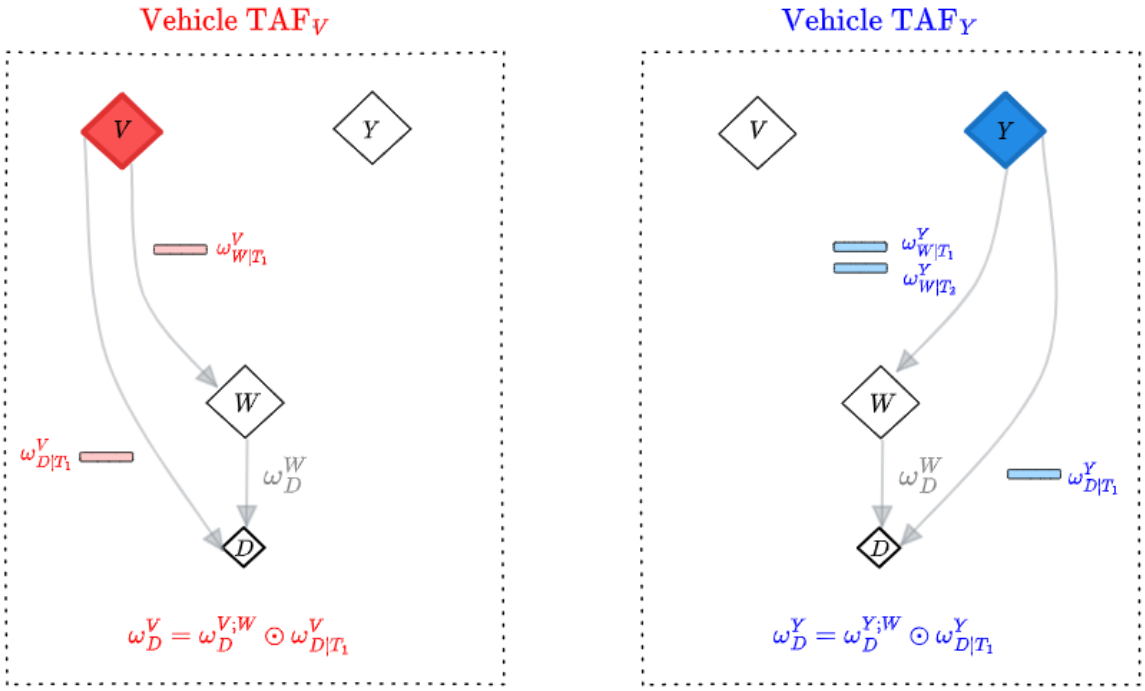


Figure 5.6: Example: two vehicles. Standalone case.

5.5 Case 3 - Distributed calculation of a trust opinion

In this section, we finally consider a scenario where the federated TAF architecture is leveraged to assess the same trust object, by TAF_A and by TAF_B. As a motivating example, imagine vehicle V and vehicle Y, in the same neighborhood, both receiving V2X data D from vehicle W. Both V and Y are interested in assessing the trustworthiness of the data D, as explained for the Vehicle TAF in Figure 5.5.

Both V and Y run the onboard misbehaviour detection service, and can produce trust evidence both relevant to the V2X data D (data-centric misbehaviour detection) and to the vehicle as source of V2X data W (node-centric misbehaviour detection). Both TAF_V and TAF_Y receive TARs to evaluate the ATLs $\omega^V_D\big|_{\mathrm{TAF}_V}$ and $\omega^Y_D\big|_{\mathrm{TAF}_Y}$, as detailed in Figure 5.6.

The interest in federating TAF_V and TAF_Y resides, in this case, in making the whole evidence available at both sides, which is beneficial because it should provide a reduction on the uncertainty on both ATLs. The most straightforward way to achieve this is by implementing federation instances so that all the atomic opinions from the trust sources can be shared. This is shown in Figure 5.7.

Sharing all the available atomic opinions would require several federation requests to take place (one for each shared atomic opinion). This has the drawback both of provoking a communication overhead (due to the multiple federation processes) and a potential increase in the complexity of both TAF_V and TAF_Y, which need to embed information about all the available trust sources. For these reasons, the federation solution that appears most favorable is the one depicted in Figure 5.8. In this case, only two federation processes take place (one from TAF_V towards TAF_Y, and one from TAF_Y towards TAF_V). Notice that the final ATLs $\omega^V_D\big|_{\mathrm{TAF}_V}$ and $\omega^Y_D\big|_{\mathrm{TAF}_Y}$ will have

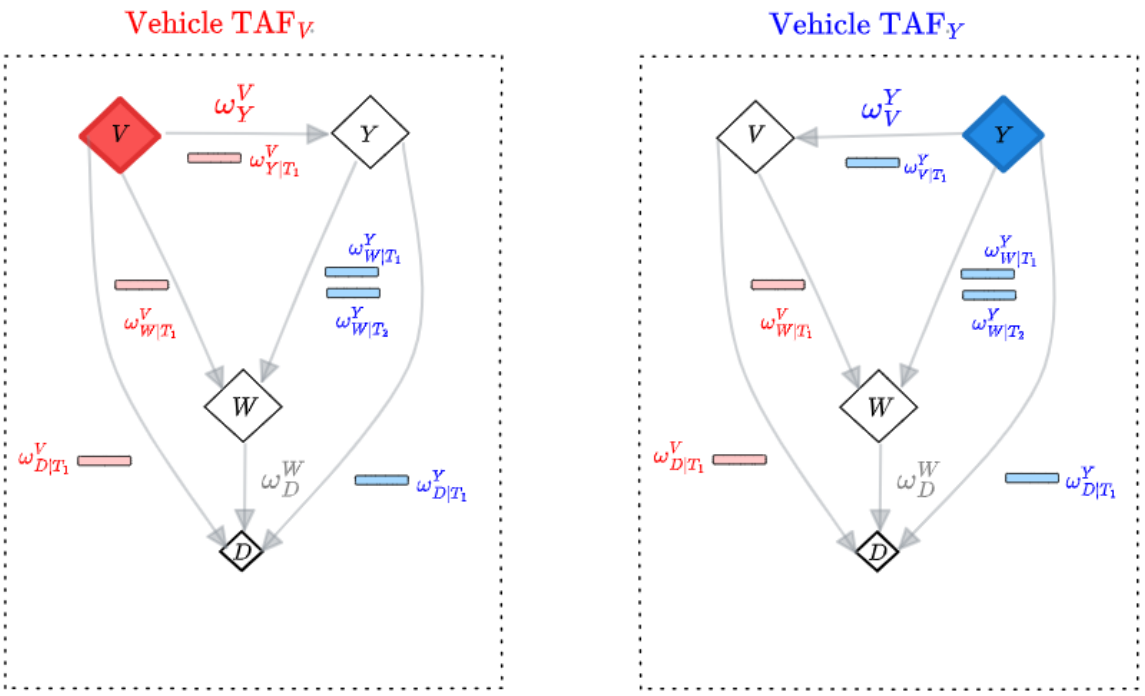


Figure 5.7: Example: two vehicles. Federation, sharing of the atomic opinions from the trust sources.

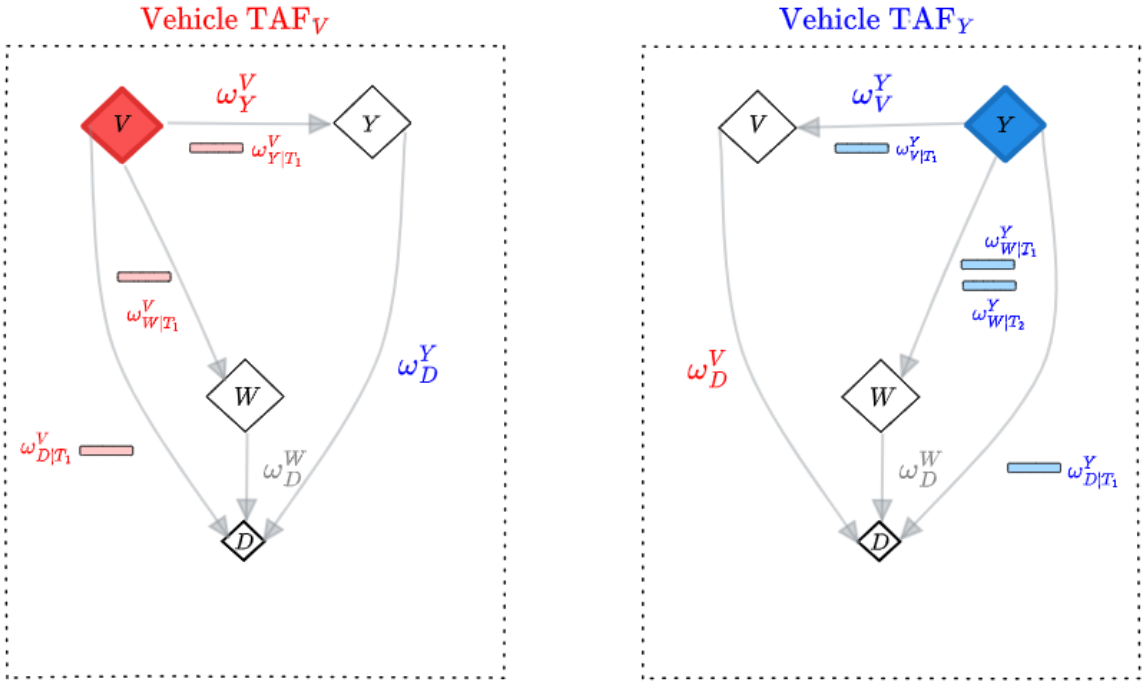


Figure 5.8: Example: two vehicles. Federation, sharing of trust opinions.

the same value only if both $\omega_Y^V\big|_{\mathrm{TAF}_V}$ and $\omega_V^Y\big|_{\mathrm{TAF}_Y}$ correspond to full belief, zero uncertainty opinions. In this case, TAF_V and TAF_Y may be understood to work on the same global trust model; and the federation principle allows a distributed evaluation of the required ATL, where half of the computation is performed by TAF_V and the other half by TAF_Y.

Note that there can be cases where evaluation of TAF_V evaluates back to TAF_Y and this might lead to (prohibited) circular evaluations that are not identifiable in any single trust model. Therefore, we only allow forward and no backward references, i.e., if TAF_V references TAF_Y then if evaluation of a trust object in TAF_Y encounters a link back to TAF_V, the evaluation would be aborted and fail. A more complex algorithm to avoid cycles can be envisioned.

Chapter 6

Digital Twin

6.1 Motivations for a Digital Twin

Traditionally, a Digital Twin is an architectural pattern where the state of a physical entity is replicated in some way in the digital space. This replication can leverage various technologies like sensors to observe the physical entity, data connections to components inside the physical entity, simulation models which represents the entity's behaviour or 3D/2D models representing the entity. Digital Twins can be used in a range of different use cases, most commonly related to system monitoring, prediction of maintenance, change management or remote operation.

In CONNECT, we want to use such a Digital Twin to support offloading the processing of Trust Assessment Requests from a TAF inside a vehicle to a MEC server. One rationale behind this use case is that Trust Models and Trust Sources may become so large that the computations performed to obtain the Trust Levels will require too much computing power for the processing capabilities available inside a vehicle. With the CONNECT architecture also taking advantage of the capabilities of Edge Computing, we consider it possible to perform those computations in a much less restricted environment while staying close enough to the vehicle in order to achieve acceptable latencies.

So in CONNECT, a digital twin of a trust assessment framework (termed TAF-DT) represents a remote replica of a vehicle's TAF, including all of its state, like trust models, trust sources, and inference capabilities which runs in a MEC device. This is similar but not identical to the definition of digital twins given above, as we do not create a model of a physical entity, but rather replicate a model of trust relationships in a second location.

The offloading capability offered by a TAF-DT could be especially worthwhile in the context of a federated TAF when a MEC server running a TAF-DT for a vehicle could query that vehicle's TAF (or rather its digital twin) without actually communicating with that vehicle. Instead, it now becomes possible to query the state of other vehicles, or trigger computations involving their trust models, without imposing additional load on those vehicles or the network. This would provide numerous advantages, for example, it lessens delays caused by higher priority computations needing to be run on a vehicle when a federated request is received or reduces the exposure of vehicles to denial of service via flooding them with federated requests.

On the other hand, it also creates new challenges to be solved, foremost the replication and synchronisation of the TAF-DT with the original TAF (also called main TAF).

In this chapter, inline with our timeline and approach presented in Deliverable D3.1, we provide

a first analysis of the TAF-DT in terms of challenges, and requirements, but also architectural foundations. A fine-grained architecture and detailed description of the TAF-DT will be presented in Deliverable 3.3.

6.1.1 Challenges

In this section, we identify the main challenges to design a TAF-DT. While we will not be able to tackle all of them in CONNECT due to limited resources and time, we still aim to provide pointers on how to resolve them.

Challenges

The core challenge will be to keep the TAF-DT's state consistent with the main TAF, ensure availability of both TAFs, while still allowing some partitioning tolerance. As shown in the CAP theorem, not all three can be guaranteed at the same time, so some compromises will have to be made.

- **Consistency of state:** While strict consistency may come with an unacceptable burden in latency and network overhead, the main TAF and TAF-DT should be eventually consistent, i.e., always converge to the same state. Also the drift of state between both should be limited. As ensuring perfect synchronization between the twin and the physical counterpart is not something we can expect to do easily, to address this challenge we plan to study how trust opinions behave when introducing small changes in the underlying trust model.
- **Availability:** A stable communication link between main TAF and TAF-DT must be maintained while respecting the intermittent nature of automotive communication. While TCP provides many of the required properties, it is not particularly suited for vehicular communication. Another problem for availability will be handovers where the vehicle moves from one cell to another. Pseudonym changes would be another challenge, however, for our practical design and implementation, this will be disregarded to contain complexity.
- **Latency:** Ensuring timely responses and communication between TAFs is a precondition to enable consistency and maintain latency requirements of applications. While this is already challenging for the standalone TAF, federated TAF and TAF-DT will have additional sources of latency. For the TAF-DT this might be caused mainly by synchronization delays. On the other hand, a MEC server querying a local TAF-DT instead of a remote TAF in a vehicle might also bring advantages in terms of latency. Note that in CONNECT, we will only be able to address some aspects of latency, as we do not have full control of the cellular network and telecommunication operator infrastructure.
- **Confidentiality of TAF:** Because the TAF-DT replicates the state of a vehicle, it also contains confidential information about this vehicle and its trust model, so we must make sure that this state does not leak outside of the TAF-DT execution environment. To achieve this, we plan to rely on trusted execution environments investigated in CONNECT in the form of Intel SGX and GRAMINE.
- **Integrity of Computation:** The TAF-DT will be tasked with trust computations on behalf of the vehicle, but it will run outside of the control of said vehicle. Any entities querying

the TAF-DT must be able to trust that the TAF-DT itself is trustworthy and its computations cannot be manipulated, for example, by a maliciously manipulated MEC server. To this end, we also plan to leverage trusted execution environments.

Depending on the targeted use cases, a Digital Twin can be implemented in different ways. For example when focusing on Asset management (eg for maintenance prediction and equipment monitoring) the bulk of it would be time series made from telemetry data captured by sensors on the physical counterpart. In other use cases, for example if the digital twin itself produces data, it could also contain a simulation model, a 3D representation of the physical counterpart etc..

As our TAF-DT represents a very special notion of a digital twin, we come up with specific challenges and requirements.

As a result, we should correctly identify the scope of the digital twin use case and derive appropriate requirements in order to then take appropriate design decisions and select the appropriate technologies to implement it.

6.1.2 Requirements

To fulfill the objectives highlighted previously we define a number on requirements on the TAF-DT

Execution of the TAF computations related to a specific TAF instance in an edge node	
Description	The TAF-DT must be able to transfer requests issued locally in a vehicle to a remote instance of the TAF and execute it there.
Rationale	This requirement implements the basic offloading functionality.

TAF computations in the edge node are based on a replica of the TAF instance state	
Description	The two TAF instances must be able to communicate and synchronize their internal state with sufficient quality.
Rationale	In order to achieve sufficiently comparable outcomes on trust computations, the state of the standalone TAF in the vehicle and the TAF-DT must remain synchronized. Strict consistency will very likely violate latency and other QoS requirements of applications, therefore, a weaker form of consistency (like eventual consistency) is aimed for. Furthermore, we assume that a event-based update mechanisms will be used that delivers events which change the state in parallel to both nodes. The update mechanism can also be opportunistic in that it uses spare communication bandwidth for synchronization.

Communication with other TAF instances inside MEC.	
Description	A TAF-DT instance must be able to interact with other TAF instances (in particular those inside the same MEC server) in order to query their opinions or publish its own.
Rationale	This requirement enables functionalities necessary to use replicated TAF instances in a federation use case.

Confidentiality of the TAF internal state inside the edge node	
Description	An edge node must not be able to read or infer the internal state of the replicated TAF instances it hosts.
Rationale	The TAF-DT replicates the internal state of a component of the vehicle. It is the responsibility of the vehicle to allow access to part of its state, therefore the TAF-DT must not disclose state beyond what the admitted TARs would reveal anyways.

Integrity of the TAF internal state inside the edge node	
Description	An edge node must not be able to modify the internal state of the replicated TAF instances it hosts.
Rationale	The TAF-DT replicates the internal state of a trusted component of the vehicle. The TAF-DT must ensure that it accurately reflects this state.

6.1.3 Implementation in CONNECT

Evaluation of results

In CONNECT, we are mainly interested in the ability of the edge to complement interactions with vehicle. The TAF-DT is mainly pursued as an illustration of these interactions as it can be queried either from a vehicle or the edge. The main research questions to be addressed by the TAF-DT in CONNECT are the following:

- **What are the synchronization constraints between the trust model of the vehicle and of the digital twin which should be considered to produce useful results?** These constraints may refer to a minimum frequency for pushing updates from the vehicle to the twin, whether it is necessary to synchronize the whole state at once or if it can be done incrementally. These results will inform us on the viable solution space for implementing an offloaded trust management solution. The trade-off between consistency of state between the two entities and added latency needs to be investigated through simulation studies.
- **Which strategies can be used to minimize the amount of data transfers to synchronise the twin and the vehicle?** For example, if we can synchronize the trust model partially, how to choose which parts will be transferred? Or can we let the twin build part of the trust model autonomously and resolve disagreements later? This may lead to identify research opportunities to explore further.
- **Is it feasible to offload the computations of trust opinions for a vehicle to a digital twin?** Based on the results of exploring the previous questions, and taking into account the state of the art regarding the performances of 5G networks, we should be able to compare the requirements for our digital twin solutions to what should be achievable. We also plan to demonstrate the offloading capability in the Slow Moving Traffic Detection use case to provide concrete feedback on the deployment.

As a result, to assess the benefit provided by the TAF-DT, we will focus our experiments on the achievable consistency between TAF-DT and original TAF opinions vs. additional latency incurred. This divergence will be measured according to different parameters like the synchronisation rate, whether the twin uses only data supplied by its physical counterpart or aggregates

data collected from V2X messages. The baseline to compare with is the case without TAF-DT where the MEC or an entity sending a TAR have to resort directly to a standalone TAF.

Limitations

In this experiment, we only replicate the TAF and not the whole vehicle as a more comprehensive digital twin would do. For example instead of having the twin compute opinions based on the raw evidences produced by the vehicle's ECUs, we will let the vehicle TAF compute the opinions and synchronize those opinions with the twin.
A limitation of our approach will be that performance analysis and measurements will have to resort to a simulation toolchain and not be integrated and measured in real settings. This is partly due to resource constraints, but also due to replicability and controllability of the experiments. The CONNECT consortium does not include any telecommunication operator which could host the edge node as part of their infrastructure, so any performance evaluation would miss the actual network performance part.

Implementation Roadmap

Regarding the planning for implementing the TAF-DT, we plan two main releases:

The first release will be scheduled for M30. It should cover at least the two main functionalities of state replication and remote execution of computations. State replication implies being able to maintain a communication channel between the vehicle and the twin and to verify that changes in the trust model in the vehicle are transferred to the twin through this channel. Remote execution of computations implies that we are able to send TARs to the twin and obtain an ATL or a TD in response.

The second release will be scheduled for M33 and include integration into trusted enclaves and TAF federation. Integration into trusted enclaves will cover the establishment of secure and authenticated channels for communication and state migration across enclaves based on the mechanisms described in D4.2. TAF federation refers to the capability offered in the Federated TAF to allow TAFs to send TARs to other TAFs, where we would also allow to query the digital twin of a TAF.

6.2 High-level Architecture of TAF-DT

6.2.1 Digital Twin reference architecture

Standards are currently being developed to assist in leveraging the Digital Twin Pattern in a system architecture. Most notably ISO/IEC 30188 aims to define reference architectures for a variety of use cases. These reference architectures define viewpoints to help different stakeholders design, implement, build or use the system. In the following, we will focus on the Foundational and Functional viewpoints as we are in the early design phases.

The Foundational viewpoint covers concerns such as "What is the Digital Twin?", "What are the implication of building this system as a Digital Twin?", "What are the benefits of using a Digital Twin?", as well as an overview of what is being twinned and how the lifecycles of the twinned

object and its digital representation interact together. This viewpoint is roughly covered by the previous section on the motivations behind this Digital Twin.

The Functional viewpoint is defined according to the following taxonomy of bases functions: Define, Execute, Monitor, Measure, Analyse, Control and Optimise. In the Design phase of the digital entity lifecycle, the proposed functions to consider are the following:

Base function	Functions
Define	• Model design • Digital thread design • Digital twin system design
Execute	• Better understanding and selection of tools, techniques and data
Monitor & Measure	• Visual design • Simulation of digital twin implementation process
Analyse	• Better understanding and analysis of physical entities
Control	• Reduce system cost
Optimise	• Realize design optimization

Table 6.1: Functions of a Digital Twin in the Design phase

6.2.2 Functional architecture

To further define the Functional viewpoint of the TAF-DT architecture, we go back to the following figure highlighting the main functions in deliverable D2.1.

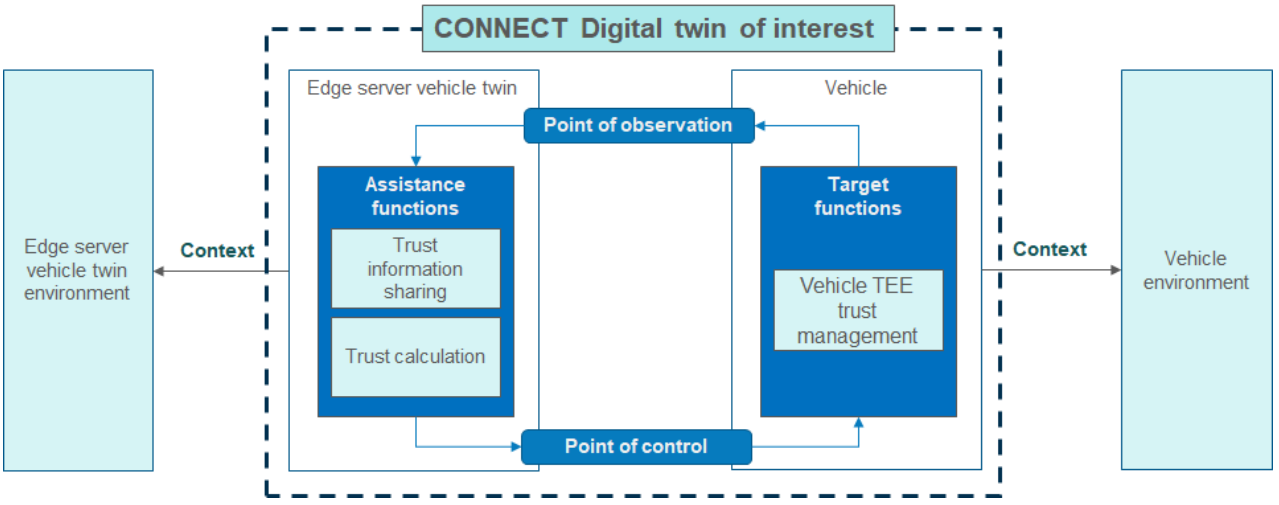


Figure 6.1: Main functions of the TAF-DT

The elements defined in this figure are as such:

Target and assistance functions

For the Target function, the Vehicle TEE trust management is the standalone TAF, i.e. the function which computes trust levels based on a trust model and trust sources.
For the Assistance functions, Trust calculation corresponds the offloading capability and Trust information sharing to exchange capabilities, for example to the Federated TAF.

Points of interaction

Interactions between the twin and the physical counterpart can occur in both direction:

- Physical to Twin:
  - ✓ Synchronisation module integrated inside local TAF. This module is used to synchronise changes in the trust model from a vehicle to its twin.
  - ✓ Sensors connected to MEC infrastructure. Used to gather information from V2X messages which could also be fed into the digital twin.
- Twin to Physical:
  - ✓ Trust computation results. These computations are triggered from requests issued by the vehicle or a remote query.

Support functions

On top of the main functions of a digital twin, we need other functions to support the deployment, execution and ensure security of the TAF-DT:

- Management by MEC:
  - ✓ Create and Destroy twin instances. The MEC infrastructure is in charge of managing the containers in which the digital twin will be hosted, to deploy them inside trusted enclaves and securely destroying them when they are no longer needed.
  - ✓ Route requests to the correct node. The vehicle does not know in advance where or how the digital twin will be deployed so the MEC will have to provide a mechanism either to route transparently requests addressed to a well-known contact point or to provide the specific contact addresses after deployment.
  - ✓ Provide simulated network between twin instances. To support communication between TAF instances, for example in federated use cases, the containers should be able to establish communication channels.
- Security Properties:
  - ✓ Confidentiality and Integrity of twin internal state provided by Gramine/SGX
  - ✓ Authentication between twin and vehicle provided by the mechanisms decribed in D4.2.

6.2.3 Allocation of TAF components

In this section we evaluate the difficulty and potential gains in offloading different parts of the TAF as part of the digital twin.

We consider the following criteria:

- The main goal of the TAF-DT being offloading, components doing heavy computation would be the one which gain most to be hosted in a digital twin.

- Synchronizing state will be one the main challenges in the TAF-DT so we want to minimize the amount of data which needs to be synchronized, as part of the state of the component. Inputs and outputs may also be taken into consideration.

From these two criteria, we derive the following characteristics:

- Computation complexity: Evaluates if the components performs heavy or light computation. The main goal of the digital twin is to enable task offloading in the MEC and this characteristic highlights which component would have the most gain to be offloaded.
- State dependency: Evaluates if the computations made by the component depend on some state maintained during its execution or if it is stateless. This characteristics are relevant because any state should be synchronized between the vehicle and the digital twin. Higher state dependency makes a component harder to put in the digital twin because of the amount of data which must be synchronized.
- State volatility: Evaluates the frequency at which the state of the component changes. This characteristic shows how a component would constrain state synchronisation between the vehicle and the twin. A higher volatility implies more stringent performance requirements for the synchronization mechanism.
- I/O complexity: Evaluates the complexity of the data which passes through the component's interfaces. This characteristic takes into account raw data size and complexity of the data structures handled. It is somewhat correlated with State dependency because previous inputs or outputs can be retained by the component as state. A higher I/O complexity on a component which is an interface of the digital twin can make it harder to implement the twin if the connection is unreliable.

By using these characteristics, the ideal candidate for being offloaded in the digital twin is a component which has high computation complexity, low state dependency, low state volatility and low I/O complexity. However, a high state dependency may be counteracted by a low state volatility because then it would not need to be updated as often.
The following table summarizes how each component of the TAF scores for these characteristics:

Component	Computation complexity	State dependency	State volatility	I/O complexity
TMM	medium	high	high	high
TSM	low	high	high	high
TLEE	high	none	none	high
TDE	low	low	low	low

Table 6.2: Complexity of integration in TAF-DT

From these results, we see that the TMM and TSM are the hardest components to replicate on the digital twin, managing both complex and volatile states while not being computationally intensive. Meanwhile, the TLEE does not need to manage an internal state but performs a lot of computations, the main issue being its dependency to complex inputs from the TMM.

An ideal configuration of components may be to replicate the vehicular TMM, TLEE and TDE in the digital twin and not replicating the TSM, relying instead on a shared TSM among replicas to

listen for evidences available from the V2X network and only synchronising evidences coming from source inside the vehicle. This should be coupled with a caching mechanisms that caches trust source state in the digital twin and which gets updated with an event-based mechanism that guarantees eventual consistency but not strict consistency. This will allow the TAF-DTs view on trust sources to lag behind to a certain but limited extend, but will make sure that the TAF-DT is guaranteed to catch up as new updates arrive. What needs to be avoided is a systematic drift of the TAF-DTs state from its twin's standalone TAF's state. Furthermore, we need to investigate how this drift will affect application quality when relying on a TAR sent to a TAF-DT instead of the original TAF. We expect here that some deviation is tolerable, as updates of Trust Sources will, depending on the type trust source, also happen with a certain delay in the standalone TAF.

Chapter 7

Risk Assessment Framework

7.1 Introduction

In the previous chapters, we have examined all the different modalities of the CONNECT Trust Assessment Framework that can enable the dynamic trust assessment of both node but also data items: From a local, standalone TAF component (Chapter 4), to enabling the collaboration of multiple TAF instances for a common trust model calculation (Chapter 5), up to the replication of a TAF component in a digital twin (Chapter 6). The common denominator of the trust assessment framework is the ability to assess trustworthiness based on monitored evidence coming from various trust sources for a particular behaviour of an asset. A behaviour can be a function of a vehicle such as the change of lane in a predefined time window or even the integrity of kinematic data. This node- and data-centric trustworthiness is essentially based on assurance claims, as modelled in the abstract IoT Conceptual Model specified in ISO/IEC 30141:2018 [18] presented in Figure 7.1. These assurance claims are derived from evidence that are collected from the assets of the target system and are directly linked with the security mechanisms, representing the assets' capabilities. In the context of CONNECT, such evidence may come, for example, from intrusion detection systems, misbehaviour detection or attestation results which contribute to the achievement of specific security requirements. The presence of these capabilities is beneficial for the evaluation of trustworthiness with respect to specific trust properties (i.e., characteristics such as safety, security, resilience, privacy, robustness).

Based on this definition of trustworthiness, the CONNECT trust assessment framework is able to evaluate the required level of trustworthiness of a particular trust object (i.e., data item or node) and eventually compute a Required Trust Level (RTL) for an entire CCAM function. Additionally, based on trustworthiness evidence (e.g., Misbehaviour Detection reports, Attestation evidence) collected from various trust sources during runtime, it is able to calculate the respective Actual Trust Level (ATL) that characterizes a CCAM function. A common facilitator for all trust calculations is the identification of most prominent threats and risks that affect the target trust properties. Hence, a comprehensive risk assessment framework plays an important role in trust assessment, as it contributes to the ATL and RTL calculations through the capture of all attack vectors across the CCAM continuum and evaluation of their impact per trust property (e.g., a specific attack vector may have a lower impact when assessing the privacy of a function but impose a critical risk when it comes to its security).

Risk Assessment in automotive is a well-defined research domain. There have been standardized frameworks that enable risk assessment in different properties of interest in mind: Threat Analysis

and Risk Assessment (TARA) [17] for security or Hazard Analysis and Risk Assessment (HARA) [16] for safety. Each of these methodologies treat risk values differently for each trust property and they are considered as standalone methodologies. In fact, such methodologies are specifically tailored to associating existing vulnerabilities and risks to a CCAM function. Especially for TARA, this is based on the assumption that the in-vehicle topology is an all-in-one system, and the OEM has already run an extensive threat analysis and has identified threats, vulnerabilities, and attack paths across the monitored topology.

However, in the scope of CONNECT, a vehicle is considered as a System-of-Systems as it consists of multiple internal elements, each of which may have its own vulnerabilities. The fact that a TARA requires an exhaustive threat analysis to be carried out a-priori, it is crucial to incorporate risk methodologies that contribute to the identification vulnerabilities and the evaluation of their impact in various levels of the in-vehicle components (e.g., firmware vulnerability, software-related vulnerability, hardware vulnerabilities). Such methodologies enable the automated and seamless association of vulnerabilities and threats with specific elements of a vehicle, thus facilitating the overall threat analysis that needs to be performed by security professionals. The initial goal of CONNECT is to explore different approaches towards combining different risk methodologies. Apart from the TARA methodology, we investigate a methodology that adopts the Common Vulnerability Scoring System (CVSS) [26] for the evaluation of the vulnerability impact into the risk quantification. This exploration ranges from evaluating how the CVSS-based methodology contributes the automation of the TARA methodology, up to how the results of the two can be converged to showcase how the overall CONNECT RA framework can transparently contribute to the trust calculations (i.e., both ATL and RTL). By providing a modular risk assessment framework, the ultimate goal is to evaluate the requirements for integrating additional risk methodologies, such as HARA.

Generalizing this vision, the CONNECT RA Engine is a risk assessment framework that is easily extensible with additional methodologies that can be tailored to specific CCAM entities (e.g., assets, functions) as well as trust properties. The aim is to provide a comprehensive, holistic framework that harmonizes the results stemming from the various supported methodologies and provides risk information for the different attack vectors across the CCAM landscape. To avoid having the same attack vector being processed by various risk assessment methodologies the CONNECT RA Engine is able to support the appropriate filtering capabilities that prohibit the double consideration of an imposed risk into the trust calculations for each trust property.

7.2 Risk Assessment Methodologies in CONNECT

Risk assessment methodologies vary widely in their approach and focus, each tailored to address specific aspects of risks within diverse domains. The challenge lies in accommodating this diversity to ensure a comprehensive and accurate analysis of potential risks. One the one hand, the Threat Analysis and Risk Assessment (TARA) is an ISO standardized methodology, commonly employed in the automotive industry. TARA is specifically designed to evaluate risks within automotive systems, considering factors such as connectivity, functionality, and safety requirements unique to vehicular environments. To carry out such an assessment, a thorough and detailed threat analysis needs to take place beforehand. On the other hand, domain-agnostic specifications, such as the Common Vulnerability Scoring System (CVSS) [26], are based on widely adopted vulnerability impact measurements that contribute to the overall risk calculations.

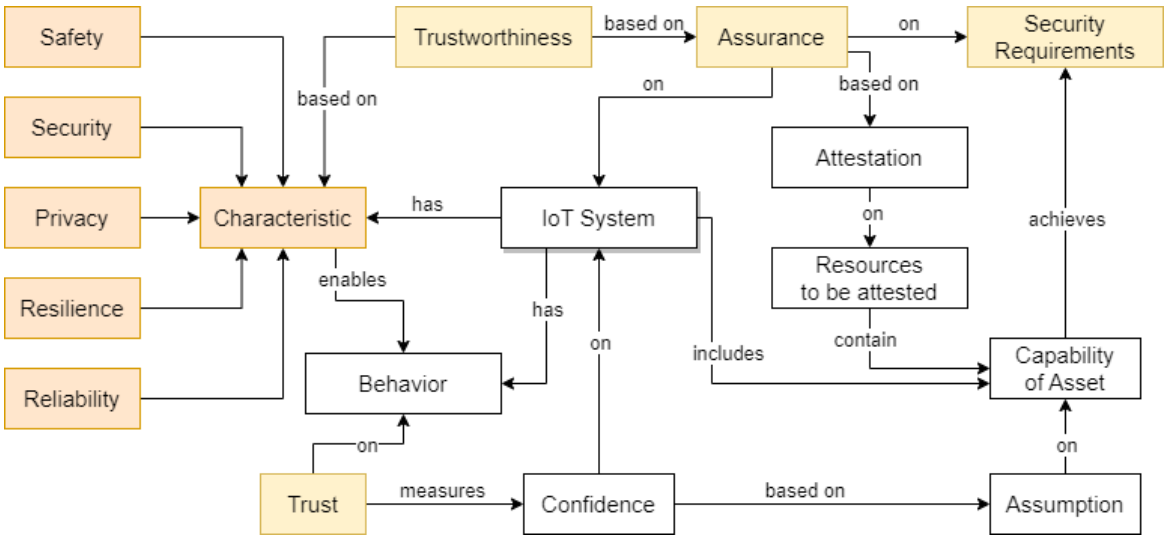


Figure 7.1: A Conceptual Model of Trustworthiness within CONNECT

It is based on vulnerabilities and threats that are well-documented in, public, repositories (e.g., CVE [3], CWE [4], CAPEC [1]) and associated with specific types of assets and specific products (e.g., CPE [2]). Such specifications enable a seamless and automated risk assessment workflow across a wide range of systems and domains.

7.2.1 Threat Analysis and Risk Assessment in Automotive

Threat Analysis and Risk Assessment (TARA) is a framework standardized by the ISO/SAE 21434:2021 [17], an international standard for cybersecurity engineering for road vehicles, including cybersecurity processes, risk management, and promotes a cybersecurity culture for road vehicles. One of the main contributions of this standard is a framework which includes requirements for cybersecurity processes and risk management, called *Threat Analysis and Risk Assessment* (TARA).

TARA is a modular method and its modules can be proceeded in any order. TARA is performed on an *Item*. [19] defines an *Item* as a ”component or set of components [...] that implements a function at the vehicle level”. This Item definition shall include at least information regarding its function, attack surface, item boundaries and operational environment.

Regarding to the TARA activities, ISO/SAE 21434 [19] defines these generic steps for an Item, that can be performed systematically from any point of the life cycle:

- *Asset identification*: Identification of assets that, if any of the cybersecurity properties are compromised, might result in an adverse scenario. This process also identifies potential damage scenarios. In this step the OEM should provide the cybersecurity properties to be examined, along with the damage scenarios that are identified in their environment. Subsequently, the OEM needs to provide a complete modeling of the assets comprising the in-vehicle System-of-Systems, including the data flows between them. Finally, each asset needs to be associated with one or more cybersecurity properties and how the compromise of each of these properties leads to the identified damage scenarios.

- *Threat scenario identification*: Determine attack scenarios that target one or more assets while threatening one or more cybersecurity properties.
- *Impact rating*: The impact of a damage scenario is measured and assigned to four categories: safety (s), financial (f), operational (o), and privacy (p), with ratings of severe, major, moderate, and negligible. The OEM needs to explicitly specify all the impact categories for each of the identified damage scenarios in scope of the asset identification process.
- *Attack path analysis*: Determining the intentional steps required to execute a threat scenario and thereby initiate a damage scenario. An attack may have many attack path possibilities. Here the OEM needs to provide an explicit list of possible ways that an adversary can pivot from one asset to another in order to cause a cascading attack in the underlying infrastructure.
- *Attack feasibility rating*: Estimates how difficult it is to carry out the considered attack. The rating corresponds to very low, low, medium, or high. From a given set of attack paths, the OEM needs to specify all the input attributes defined in the TARA methodology to determine the attack feasibility rating, namely the Elapsed Time, the Expertise, the Knowledge, the Window of Opportunity, the Equipment.
- *Risk level determination*: It is a value calculated taking into account the impact rating and attack feasibility rating. The value ranges from 1 to 5, with 5 representing the highest risk. [19] uses as example for quantification of risk the equation $R(I, F) = 1 + I \times F$, where F and I stand for Feasibility and Impact parameters respectively. An example on how the calculations of the risk values are carried out for each $(threatscenario, damagescenario)$ tuple are presented in Section 7.7. However, the risk equation can vary between manufacturers and there is not a standardized equation in the context of TARA. Having that in mind, and in the context of modularity in the CONNECT RA Engine, the end most goal is to find the harmonization and intelligence layer that enacts upon the different risk quantification methodologies. This enables a transparent and robust way for the CONNECT RA Engine to provide the necessary risk analysis for the trust calculations.
- *Risk treatment decision*: Describes the chosen treatment strategy for an identified risk. It can be done to prevent, mitigate, share, or retain the risk. In the case of mitigation, the OEM needs to provide the security controls that can be applied in the underlying assets and how these controls have an effect on the identified threat scenarios and the overall risk calculations.

The TARA methodology is a standardized risk assessment framework tailored to the automotive domain that enables security administrators to associate vulnerabilities and attack scenarios to functions. However, the breakdown of the TARA methodology presented above highlights the exhaustive and detailed threat analysis that needs to be carried out by the OEM in advance. This challenge is alleviated through the adoption of other risk assessment methodologies that, even though they can be domain agnostic, focus on the association of systems with vulnerabilities and the calculation of their impact.

7.2.2 CVSS-based Risk Assessment Methodology

This subsection introduces the CVSS-based risk assessment methodology, a comprehensive risk assessment methodology that is tailored to embedded systems. While domain agnostic, the

methodology is capable of capturing the risk of complex System-of-Systems environments as the ones comprising the in-vehicle topology (e.g., ECUs, sensors, Zonal Controllers, services). It is based on a well-adopted specification for conducting risk assessment, namely the NIST Special Publication 800-39, *Managing Information Security Risk: Organization, Mission, and Information System View* [24], that identifies the four main steps of the risk management process as well as the communication interfaces to enable the process to work effectively. According to the specification, the lifecycle consists of the following four steps

- **Risk Framing**: Establishing a context for risk management, outlining strategies for assessing, responding to, and monitoring risks transparently and explicitly within organizational boundaries.
- **Risk Assessment**: Identifying threats, both internal and external, vulnerabilities, potential harm, and the likelihood of occurrence, resulting in a determination of risk level.
- **Risk Response**: Developing, evaluating, and implementing consistent organizational responses to identified risks in alignment with organizational risk tolerance.
- **Risk Monitoring**: Continuously evaluating the effectiveness of risk responses, identifying changes in organizational systems and environments, and ensuring implementation of planned risk responses to meet security requirements.

Risk Quantification

The NIST SP 800-30 specification [25] focuses on the second step of the risk management lifecycle - i.e., the Risk Assessment. It comprehensively outlines the various risk factors pivotal in determining the magnitude of a risk. Among these factors are threat, vulnerability, impact, and likelihood. These elements collectively serve as the foundation for establishing a robust risk quantification methodology. The remainder of this subsection introduces a risk quantification methodology that combines a subjective threat likelihood parameter with a systematic vulnerability impact measurement based on the Common Vulnerability Scoring System (CVSS) methodology.

Threat Probability

According to [25], a *threat* is any circumstance or event with the potential to adversely impact organizational operations and assets, individuals, other organizations, or the Nation through an information system via unauthorized access, destruction, disclosure, or modification of information, and/or denial of service. The **likelihood** that a threat may materialize is a variable that depends on a number of parameters, as for example the infrastructure's inherent characteristics. As a result, defining a threat likelihood is an inherently subjective process, per individual infrastructure, threats and needs, that can be carried out by the OEM security administrator. Table 7.1 presents a qualitative and semi-quantitative classification of the threat likelihood.

Qualitative Values	Semi-Quantitative Values		Description
Very High	96-100	10	Adversary is **almost certain** to initiate the threat event.
High	80-95	8	Adversary is **highly likely** to initiate the threat event.
Moderate	21-79	5	Adversary is **somewhat likely** to initiate the threat event.
Low	5-20	2	Adversary is **unlikely** to initiate the threat event.

Very Low	0-4	0	Adversary is **highly unlikely** to initiate the threat event.

Table 7.1: Likelihood of a threat occurring

Vulnerability Impact

In addition, NIST defines the term *vulnerability* as a weakness in an information system, system security procedures, internal controls, or implementation that could be exploited by a threat source. Associating vulnerabilities with specific assets enables security administrators to evaluate their impact in a systematic manner. Specifically, the Common Vulnerability Scoring System (CVSS) [26] developed by the Forum of Incident Response and Security Teams (FIRST) [6] is a globally recognized standard for assessing and rating the severity of software vulnerabilities. CVSS provides a structured framework for security professionals to objectively evaluate the potential impact and exploitability of vulnerabilities, aiding in prioritizing remediation efforts and allocating resources effectively. Annex A summarizes the CVSS attributes as defined in the 3.1 version of the specification. According to the specification, there are three core metrics, namely the Base, the Temporal and the Environmental metrics.

The Base group (Mandatory Metric) represents the fundamental features of vulnerabilities that remain consistent over time and across user environments. It is further analyzed to two submetrics the exploitability and the impact metric. The exploitability metric describes the technical feasibility of the vulnerability, while the impact reflects the results that a successful exploit will have and depicts the influence on the affected component.

The exploitability metric is further described by:

1. the *Attack Vector (AV)*, which indicates the level of access of the attacker, hence it can take four values: Physical (P), Local (L), Adjacent (A) and Network (N), moving to the one with the most requirements needed, to the one that can be achieved even remotely.
2. the *Attack Complexity (AC)*, this metric receives values Low (L) and High (H), depending on the preparation needed by the attacker in order to exploit.
3. the *Privileges Required (PR)*, which accepts values None (N), Low (L) and High (H), depending on the level of privileges required by the attacker to launch the attack, with low signifying basic user capabilities and high administrative privileges.
4. the *User Interaction (UI)*, referring to the interaction with a human user, receiving values None (N) and Required (R) if a user is needed.

On the other hand, the impact metric is further described by the CIA triad:

1. Confidentiality,
2. Integrity,
3. Availability.

Each of the CIA requirements received a score among the values High (H), Low (L) and None (N), depending on the level of intrusion that the attacker can successfully reach.

$$ISS = 1 - [(1 - Confidentiality) \times (1 - Integrity) \times (1 - Availability)]$$

The Temporal class (Optional Metric) reflects vulnerability characteristics that may alter over the course of time. Temporal metrics are further described by three metrics:

1. the *Exploit Code Maturity (E)*, which describes the feasibility to exploit, defined by Not Defined (X) due to lack of information, High (H) maturity with automated tools available, Functional (F), Proof-of-Concept (P) and Unproven (U).
2. the *Remediation Level (RL)*: which considers the possible patches for the identified vulnerability, defined by Not Defined (X), Unavailable (U), Workaround (W), Temporary Fix (T) and Official Fix (O).
3. the *Report Confidence (RC)*: which illustrates the certainty over the technical details and existence of vulnerability, described by Not Defined (X), Confirmed (C), Reasonable (R) and Unknown (U).

$$TemporalScore = \{mathitRoundup(BaseScore \times E \times RL \times RC)$$

The Environmental category (Optional Metric) demonstrates vulnerability characteristics that are unique to a user's environment. Environmental metrics are further described by the Security Requirements (SR) and adapt the CIA triad per the specific IT environment requirements (hence CR, IR and AR). Their score ranges among the values High (H), Low (L), Medium (M) and Not Defined (X), depending on the level of intrusion that the attacker can successfully reach.

$$MISS = Minimum(1 - [(1 - ConfidentialityRequirement \times ModifiedConfidentiality) \times (1 - IntegrityRequirement \times ModifiedIntegrity) \times (1 - AvailabilityRequirement \times ModifiedAvailability)], 0.915)$$

The Basic metric generates a score ranging from 0 to 10, which can then be revised by the Temporal and Environmental metrics. A compressed textual representation of the values utilized to calculate a CVSS score is also known as a CVSS vector string. A vulnerability, apart from the CVSS score can be described as a vector, e.g., CVSS:3.1/AV:N/AC:L/PR:H/UI:N/S:U/C:L/I:L/A:N.

Individual and Cumulative Risk Levels

Based on the aforementioned risk factors it is possible to provide a host-based risk assessment. Each asset is linked to a series of vulnerabilities, with each vulnerability corresponding to a distinct threat. Based on the threat likelihood and vulnerability impact modeling presented above, it is possible to calculate the individual risk level for each tuple $(Asset, Vulnerability, Threat)$ as follows - qualitative and quantitative scales are listed in Table 7.2:

$$IRL(Asset_i, Vulnerability_j, Threat_k) = ThreatLevel \times VulnerabilityImpact$$

Note: Computing an aggregated risk value for an asset is inherently context-dependent, emphasizing that there is no one-size-fits-all approach. Different approaches consist of taking the maximum IRL value, computing a weighted average or maintaining risk matrices, categorizing threats or vulnerabilities based on their likelihood and impact. It is up to the security administrator to determine what is the best approach for their application domain or scenario.

IRL Qualitative Values	IRL Quantitative Values
Very High	96-100
High	80-95
Medium	21-79
Low	5-20
Very Low	0-4

Table 7.2: Risk values in CVSS-based methodology

This type of risk assessment methodologies measures the risk of an asset without taking into consideration the dependencies and the risks introduced by neighbouring assets. A holistic risk assessment framework should be to take into consideration the attack paths that an adversary may leverage to cause harm in a system. In this methodology, the interdependency graph of an asset topology is adopted. This allows security administrators to express any type of interconnection (e.g., physical, logical, network, flow of data) within the monitored infrastructure. Consequently, any cyber-physical ecosystem can be modeled and visualized as a Direct Acyclic Graph (DAG), based on the notion of the interdependency graphs. Based on this data model, it is possible to define an attack path starting from an entry point and through a set of pivoting steps are able to reach the target asset, thus forming a chain of assets. Each pivoting step allows an adversary to exploit a single vulnerability in an asset. This can result in multiple attack paths targeting the same set of assets, with adversaries exploiting different combinations of vulnerabilities.

The details for the design and development of the attack path analysis are presented in D3.3 [14]. This will eventually lead to the realization of the Cumulative Risk Level (CRL) that captures not only the individual risk level of an asset, but also the incoming risk imposed by neighbouring assets in the topology. The quantification of the risk level is critical to trust calculations. To this extent, it is essential to identify and fine-tune the risk assessment framework so that the imposed required trust level can be attained by all related assets in a trust model. For example, if the maximum risk value of a high-risk asset is used for the RTL calculations, this could result in lower-risk assets being unable to meet the risk threshold during runtime.

By inspecting the risk results of both the TARA methodology (i.e., Table 7.15) and the CVSS-based methodology (i.e., Table 7.2) it is clear that there are plenty of similarities between the two risk scales. The fact that both qualitative scoring systems are 5-tier enables a common understanding over the risk levels. In parallel, these two methodologies provide risk analysis on different layers of the CCAM landscape. Hence, one of the key envisions of the CONNECT RA framework is to allow the harmonization of the risk results which leads to a richer threat analysis. This enables a more accurate RTL calculation and an optimal association of trustworthiness evidence that need to be collected in scope of the ATL calculations during runtime.

7.3 Functional Specifications of CONNECT RA Framework

Table 7.3 summarizes the functional requirements of the CONNECT Risk Assessment (RA) Engine. Given that one of the main goals of the CONNECT RA Engine is to automate the risk assessment framework to the maximum possible extent, the title of each requirement is labelled as automated, semi-automated or manual. These requirements capture all the mandatory characteristics that the CONNECT RA Engine should provide to support the calculation of the values that are necessary for the trust assessment of the CONNECT ecosystem, namely the Required Trust Level (RTL) and the Actual Trust Level (ATL).

ID	Title	I want to $< Action >$	so that $< Reason >$	Description
FR_RA_1	Modeling assets (Automated)	model various types of assets (i.e., node and data items)	I can capture all the entities that characterize the CCAM continuum	In the context of the OEM and Service Provider threat analysis, users should have the means to specify the CCAM entities as well as the involved stakeholders participating in the various CCAM functions. Through this specification, users should have a granular way of expressing the CCAM entities that the in-vehicle topology may have, while maintaining the necessary level of abstraction when required. Finally, the CONNECT RA Engine has the capabilities of capturing all types of assets within the in-vehicle topology (e.g., ECU devices, sensors, CCAM software, data).
FR_RA_2	Modeling asset relationships (Automated)	capture the interdependencies among the various assets	I can model the component diagram and data flows that characterize the asset graph.	It is essential that the CONNECT RA Engine is able to model all logical, network and data flow relationships between assets of the CONNECT ecosystem. In addition, to capturing dynamic relationships (i.e., and consequently their imposing risk), the CONNECT RA Engine should allow the expression of different vehicles that are present in the local trust model of a vehicle for a short period of time (e.g., neighbouring vehicles can be considered as a black box to capture these data flows). All these relationships enable the construction of component diagrams and flow diagrams that are necessary for the execution of the risk assessment methodologies that contribute to the calculations of the RTL and ATL values. Each service may have multiple trust models (Section 4). Hence, in the CONNECT RA Engine each service can be represented by multiple asset graphs to encompass the different trust properties and scopes (i.e., data-centric or node-centric).
FR_RA_3	Visualize asset graph (Automated)	visualize the entire asset cartography	I can execute the necessary threat analysis for the risk assessment methodologies	The visualization of an asset graph describing a specific trust model and scope provides the users (i.e., OEM, security administrators) the capabilities to provide their input based on which a more detailed analysis is performed in the context of the target risk methodology. For instance, in the case of TARA methodology, users are able to have a complete overview of the asset graph and identify all the attack paths and threat scenarios in an efficient manner.
FR_RA_4	Modeling trust models (Semi-automated)	capture different trust properties in the service graph chain	I can have a better overview of the relationships and nodes that need to be assessed per trust property.	According to the conceptual model presented in Section 7.1, different assets may have multiple capabilities and, thus, participate in various services. In parallel, a service may have various trust models depending on the scope and properties that need to be assessed in the context of the trust assessment framework. Hence, the CONNECT RA Engine should be able to capture all these different asset graphs that enable a fine-grained risk analysis based on the TARA methodology. From the running example in Section 7.7, the asset topology of Figure 3.1 may result in different asset graphs: one for a service that uses GNSS data and one for another that uses the rest of the sensor data.

ID	Title	I want to $< Action >$	so that $< Reason >$	Description
FR_RA_5	Up-to-date open intelligence (Semi-Automated)	have access to the latest open intelligence information available related to threats and vulnerabilities	I can associate them with the monitored assets and evaluate their impact on the risk assessment.	In scope of the threat analysis, it is essential that the CONNECT RA Engine has access to the latest open intelligence information from publicly available repositories (e.g., National Vulnerability Database [23]). This enables an accurate identification of the security posture of the monitored assets. Apart from publicly available vulnerabilities and threats, the knowledge base should be enhanced with information of cybersecurity threat intelligence information associated with connected vehicles (e.g., UNECE WP29 regulation [8]). This requirement can be achieved in an automated manner. However, it is marked as "semi-automated" as it acknowledges the challenge of associating security controls with specific vulnerabilities. For instance, a Control Flow Integrity (CFI) attestation protocol is able to address memory-related vulnerabilities such as buffer overflows. Such vulnerabilities can be classified in the "Tampering" STRIDE class. By making the association of the CFI security measure to the "Tampering" STRIDE class, it is possible to provide an initial proposal for mitigation controls for any new vulnerability that is classified with the same threat type. In any case, the security administrator should monitor this association and adjust it if necessary, based on their experience and the topology characteristics.
FR_RA_6	Identifying attack paths (Automated)	identify the attack paths comprising vulnerabilities that allow the propagation of attacks across multiple assets	I can capture the cascading effects that enable an adversary to pivot from one asset to another through the exploitation of the associated vulnerabilities.	Risk methodologies that focus only on the identification of a host-based risk are limited and cannot detect the incoming risk from neighbouring assets. Hence, the CONNECT RA Engine needs to support the construction of attack paths to enhance the risk analysis. In addition, one of the main pillars in scope of the TARA methodology is the threat scenario identification, which focuses on determining the attack scenarios that target one or more assets. In this regard, the CONNECT RA should be able to provide the optimal quantification methodology for the Cumulative Risk Level (CRL) to support the automation of the Attack Feasibility Rating and, consequently, support a more accurate calculation of RTL and ATL values. However, it needs to be stated that the CRL quantification should be achieved without imposing additional constraints that are difficult to be met during runtime.
FR_RA_7	Continuous and dynamic risk assessment (Automated)	perform risk assessment periodically as well as asynchronously	I can have the latest risk values for the entire topology right after an incident takes place.	It is required that the CONNECT RA Engine supports the continuous execution of risk analysis so as to ensure the fresh risk information made available to the trust calculations. This requirement stems from the need to capture the runtime indications of risk which may lead to an increase in the threat probability of specific assets. In the context of CONNECT, indication of risk can be the reception of failed attestation evidence - reported in the DLT but this could be extended to other sources such as intrusion detection system events. In addition, it is important that risk assessment tasks are triggered when new vulnerabilities are inserted into the system by the users or when security incidents are detected (e.g., through monitoring systems in the topology such as the misbehavior report or other Security Information and Event Management (SIEM) systems).
FR_RA_8	Modeling security controls (Manual)	associate assets with security controls that address - fully or partially - specific risks	I can enable the calculation of impact in the risk analysis when optimal sets of mitigation mechanisms are in place.	The CONNECT RA Engine needs to model different types of controls and signal their effectiveness over the risk of the assets that have put them into effect. In addition, as specified in the TARA methodology, one of the risk assessment steps, namely the risk treatment decision step, allows users to express the way of handling risks through four distinct options: Avoidance, Reduction, Sharing/Transferring, Acceptance of risk.

ID	Title	I want to $< Action >$	so that $< Reason >$	Description
FR_RA_9	Modeling mitigation strategies (Manual)	define a set of security controls for the various assets in the topology into a mitigation strategy	I can re-perform the risk analysis and evaluate its effectiveness in the overall risk of the topology.	This requirement provides additional benefits to the risk analysis that is taking place in design phase as it provides insight on the risk fluctuations when a set of security controls are in place in the CONNECT monitored topology. These calculations are taking place beforehand, though they allow the ATL calculations to adjust the contribution of the risk values depending on the security controls that are into effect. For example, the risk level of an asset in a trust relationship may vary depending on the runtime attestation protocols in effect. The level of impact of such a security control in the risk of the asset should be calculated beforehand by the CONNECT RA Engine and shared with the in-vehicle TAF for the ATL calculations.
FR_RA_10	Dynamic recalculation of RTL (Semi-Automated)	re-calculate the overall risk of the topology (representing a service graph chain) when new vulnerabilities may be discovered from any monitored trustworthiness evidence (e.g., evidence from a "*failed*" attestation report)	I can re-calculate the RTL and communicate it across the TAF instances.	Re-calculation of RTL values is needed when new indications of risks are identified for any asset of the target service graph chain. According to the overall CONNECT framework, when an attestation protocol fails (as a source for providing trustworthiness evidence), the Attestation Integrity Verification (AIV) component is responsible for sharing the failed attestation traces to the CONNECT DLT. This enables the authorized OEM to inspect them and investigate the issues that led to this incident. The aforementioned procedure may result in the identification of new, probably zero-day, vulnerabilities. The CONNECT RA Engine should enable users to express such vulnerabilities in the knowledge base and trigger new risk assessments. Consequently, the new risks are likely to impact the RTL for the affected services and needs to be shared with the TAF running instances (i.e., across the vehicles and MEC infrastructure).
FR_RA_11	Automation of risk assessment (Automated)	provide an automated and seamless risk assessment analysis	I can achieve the maximum level of automation on the calculation of both ATL and RTL.	One key characteristic of the TARA methodology is that it requires heavy threat analysis to be carried out in advance. Security administrators need to provide a thorough analysis to model the asset topology, the associated threats and their impact on identified damage scenarios. On top of that, they have to specify the possible attack paths (i.e., threat scenarios) that can occur in the topology. In realistic cases, dealing with the multitude of attack paths related to a vast and complex in-vehicle topology, including numerous sensors, devices, services and interconnections can be a challenging and time-consuming task. Consequently, the CONNECT RA Engine should achieve the maximum level of automation that enables a seamless risk analysis, while also allowing for adjustments and parameterization by the security experts.
FR_RA_12	Convergence of risk results (Automated)	converge outputs from both qualitative and quantitative risk quantification models	I can provide a more comprehensive risk assessment for the asset graph of a CCAM function, tailored to the set of trust properties of interest.	Even though in the context of CONNECT we leverage the TARA and CVSS-based RA methodologies for the threat analysis and risk quantification, the modularity feature of the CONNECT RA Engine envisions to enable the inclusion enriched methodologies (e.g., HARA [16]) potentially better tailored to the risk analysis of a service graph chain with respect to a trust property of interest. Having the maximum level of convergence allows for a transparent risk assessment framework where different results can be compared across methodologies based on a common scale (i.e., either qualitative or quantitative). Of course, the process of converging results of different risk assessment methodologies may lead to loss of information. This is a trade-off that the security administrator needs to take into consideration when performing its analysis in scope of the trust level calculations during design phase.

Table 7.3: Functional specifications of the CONNECT Risk Assessment Engine

7.4 CONNECT RA Conceptual Analysis

This section presents the internal architecture of the CONNECT Risk Assessment (RA) framework. Specifically, it starts from a high-level overview of the overall CONNECT RA framework (and its positioning in the defined reference architecture [13]). Subsequently, it presents the CONNECT RA architecture, where all the internal components and their interconnections are specified. Next, the key characteristics of each component are analyzed, while a high-level description of the sequence of actions among them is also described. The OLISTIC risk assessment platform of UBITECH has been used as the basis for the CONNECT RA Engine. It supports the CVSS-based methodology presented in Subsection 7.2.2. However, the goal is to extensively enhance the initial artifact to accommodate a modular risk assessment framework that meets the functional requirements presented in Subsection 7.3.

7.4.1 High-level Flow of Actions

In the previous subsection, we focused on how the integration of risk assessment methodologies - like the CVSS-based methodology presented in Section 7.2.2 - is capable of softening the required information that need to be provided by the OEM in the context of TARA methodology. This section presents the CONNECT RA Engine, not as a standalone component, but as a integral part of an overall trust assessment framework. Specifically, this section showcases how the risk assessment methodology enables the RTL calculations and enables the initialization of the ATL that is computed during runtime.

Figure 7.2 illustrates the position of the CONNECT RA Engine in the entire ecosystem. It provides a detailed sequence of actions starting from the initial threat analysis and the risk calculation to the realization of the RTL values for the defined trust models and the ATL calculation that take place during runtime.

The first step for the realization of the risk assessment, and consequently the setup for the trust assessment framework, is the threat analysis. In this step, OEM and security administrators need to provide all the necessary information for their CCAM services. This information includes the asset graph comprising of all the devices, services and data items that are participating in each service as well as the various interdependencies among them. In addition, to meet the TARA methodology requirements, the threat analysis also, involves the identification of the damage scenarios that affect the monitored infrastructure and the impact that such scenarios pose on associated cybersecurity properties. Finally, during the design time, users can provide security controls in an effort to mitigate the imposed threats on the asset topology. Hence, the risk assessment process supports the evaluation of the effectiveness of security controls through the calculation of the updated risk values assuming that the controls are enforced in the topology.

The risk value computed within the CONNECT RA Engine is sent to the Trust Assessment Manager (TAM). Part of the responsibilities of the TAM component is to calculate the RTL values for each trust model and publish all the necessary Trust Models and Trust Policies in to the CONNECT DLT. Part of the Trust Policies is related also to the necessary attributes (e.g., trust sources) and risk values per security control for the ATL calculations to take place during runtime.

Once all this information is available to through the CONNECT DLT, all running TAF instances can download it and initialize their trust model manager instance. All this configuration takes place in

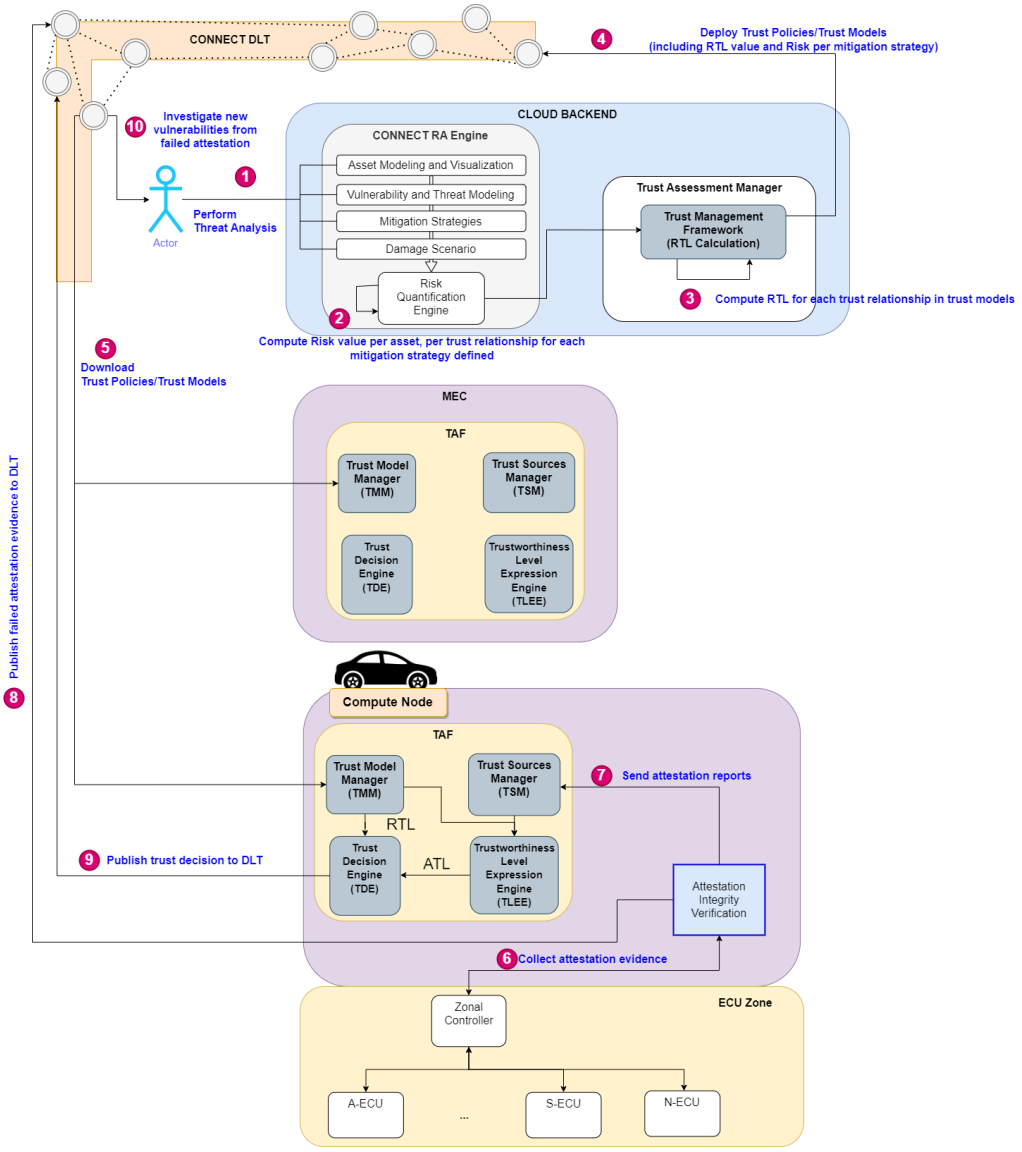


Figure 7.2: Risk Assessment Engine in the CONNECT Architecture

the context of the design-phase of the CONNECT-enabled topologies (i.e., either on the MEC or in-vehicle).

The following steps capture how the runtime operation of a CONNECT in-vehicle topology is able to provide valuable feedback in scope of the continuous risk assessment and consequently the RTL calculations. When a Request For Evidence is triggered by a standalone TAF component, it is possible to request various trustworthiness evidence, one of which is attestation evidence. In this case (step 6) the Attestation Integrity Verification component launches the requested attestation protocols and collects evidence from the underlying ECU zone. This evidence is sent in a secure fashion to the Trust Source Manager (step 7). It is worth noting that in the case of failed attestation evidence, the AIV is responsible for publishing the evidence to the DLT. This enables authorized entities (e.g., specific OEM) to analyze the failed attestation evidence (e.g., traces). Once all the necessary information has been collected, the Trust Decision Engine (i.e., TDE) in the standalone TAF - or a Digital Twin TAF in the context of task offloading - computes the trust decision based on the corresponding RTL and the computed ATL. This decision is securely published to the CONNECT DLT to enable an immutable way of keeping track of the provenance

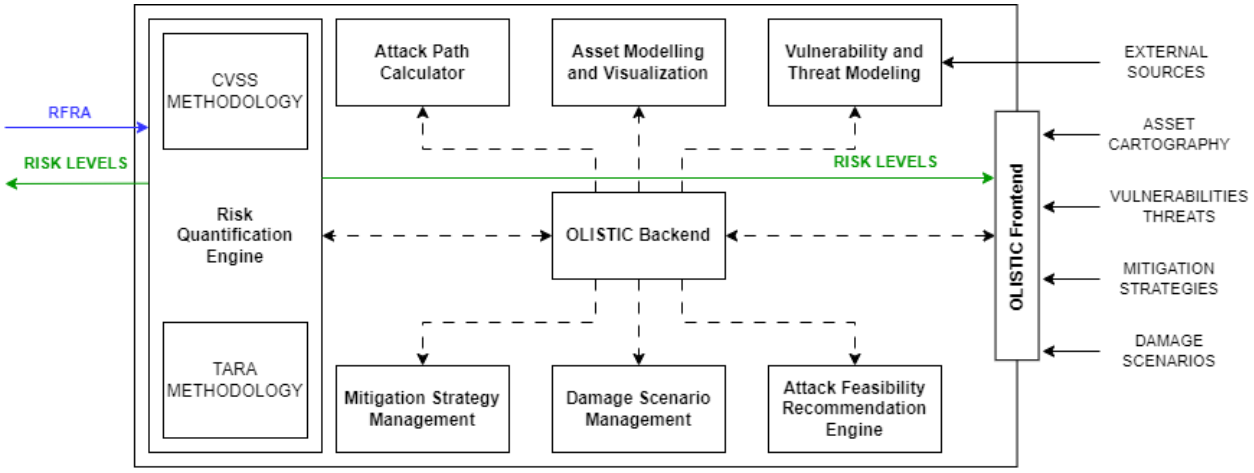


Figure 7.3: CONNECT RA Architecture

of each vehicle, or MEC.

Eventually, all these runtime stimuli can help the security administrator perform a dynamic threat analysis with runtime evidence. Specifically, in the case of failed attestation evidence, authorized users may be able to detect new vulnerabilities (e.g., zero-day vulnerabilities) or observe that despite of a control enforcement, a specific vulnerability still persists. This allows users to update the initial threat analysis, which triggers a new risk assessment process. This concludes the risk assessment workflow in the CONNECT framework and demonstrates its impact both during design phase but also during runtime.

7.4.2 Internal Architecture

Figure 7.3 illustrates the internal architecture of the CONNECT RA Engine, including the required input and the expected output interfaces. The execution of a new risk assessment task is achieved through a new Request For Risk Assessment (RFRA). This can be triggered either manually (e.g., through an action performed by a security administrator) or automatically (e.g., through a periodic, scheduled task). The result is a representation of the risk report (e.g., in the form of a risk graph, or a table of results) shared with the intended entities, namely the security administrator and the Trust Assessment Framework components. The CONNECT RA Engine supports diverse methodologies such as the TARA and CVSS-based methodologies.

To enable the execution of these risk methodologies, it is assumed that a thorough threat analysis is performed over the infrastructure that is to be monitored. This includes information about vulnerabilities and threats coming from external sources or from the security administrator (i.e., OEM). The realization of the asset cartography is also required to depict all the assets (i.e., hardware, software and data) and their interdependencies (i.e., physical, logical, data flows). Part of the initial threat analysis is related to the TARA methodology, namely the definition of the damage scenarios in the monitored domain as well as their association to the identified vulnerabilities per asset. Finally, it is also required that security administrators define the various mitigation strategies that enable the CONNECT RA Engine to assess the effectiveness of specific security measures that can be enforced in the asset topology.

7.4.3 Component Analysis

This section describes each of the internal components separately of the CONNECT RA Engine. Each of the presented components serve at least one of the functional requirements presented in Table 7.3. The main focus of this section is to delineate the scope and the requirements that each of the components meet. The technical implementation details of each component (i.e., including screenshots of the graphical user interface) are reported in D3.3 [14].

OLISTIC Backend and Frontend

The OLISTIC backend offers the necessary APIs to orchestrate all the backend operations of the engine and works in synergy with the frontend to offer the necessary functionalities to the security analyst. The backend component is also responsible for performing the access control rules that allow only authorized users to use the available API. Finally, it provides the necessary interface for participating (i.e., publishing and subscribing) to Kafka topics that enable the asynchronous consumption of security incidents reported by other CONNECT components. The OLISTIC frontend offers an interactive dashboard which is used for visualizing the digital representation of the cyber-physical environment. Of course, this dashboard offers a plethora of operations that can be performed, ranging from simple asset addition/editing/deletion to the creation of attack scenarios, management of vulnerability and threat profiles of assets, consideration of controls and mitigation actions, the execution of the risk assessment and many others.

Asset Modelling and Visualization Component

This component is responsible for modelling the list of assets that comprise the monitored CCAM ecosystem. In fact, this component is crucial for the realization of the CONNECT RA Engine as it addresses the first four functional requirements presented in Table 7.3. We consider an asset any entity, tangible or intangible, that participates in the ecosystem and carries a level of risk. To address the first functional requirement, an abstract notion of the term asset is adopted to represent hardware assets (e.g., in-vehicle sensors and devices), services running on top of them (e.g., image processing, obstacle detection system) or even data flowing across the in-vehicle topology (e.g., camera data). Apart from a uniquely identifiable name in the topology, security administrators can provide additional information such as the level of criticality of each asset in the domain (i.e., business value). In addition, it is possible to provide the Common Product Enumeration identifier, if applicable. This facilitates the seamless and automated association of the assets with vulnerabilities and threats. The CONNECT RA Engine also allows for an arbitrary set of attributes in a key-value format.

Moreover, following the conceptual model of ISO/IEC 3041:2018, assets consist of capabilities that allow them to participate in various functions/services. Each of these functions is modeled as an asset cartography comprising of a set of assets and a set of interdependencies between them. The relationships among assets can express various types of interdependencies ranging from network/physical connections, software dependencies or even data flows. An initial, yet not definitive, list of asset relationships is presented below:

- *IsConnectedTo*: to express network connections between hardware assets (e.g., ECU-1 is connected with an ethernet cable with Zonal Controller 2),

- *IsUsedBy*: to express logical dependency among assets (e.g., the GPS navigation system is used by the vehicle's onboard computer to provide real-time directions to the driver),
- *IsProcessedBy*: to express the fact that a piece of data is accessed by an asset (e.g., Sensor data collected from the vehicle's cameras is processed by the autonomous driving software to identify and react to objects on the road),
- *isLocatedIn*: to express geospatial dependency among assets (e.g., the ECU responsible for managing the vehicle's infotainment system is typically stored within the dashboard),
- *isStoredOn*: to express the fact that a piece of data is stored on an asset (e.g., the vehicle's diagnostic logs are stored on the in-vehicle computer)
- *isInstalledOn*: to express a dependency between software and hardware assets (e.g,. The lane-keeping assist software is installed on the ECU-1).

The CONNECT RA Engine offers a graphical user interface based on which the security administrator is able to manage the asset cartography. Firstly, the tool supports the visualization of the data flow diagram for each function and the inspection of the list of assets defined in the monitored infrastructure. Finally, to support the steps of the TARA methodology, the Asset Modelling and Visualization component, also, accommodates for the association of assets with specific damage scenarios and their impact in the context of specific trust properties.

Vulnerability and Threat Modelling Component

The association of assets with their vulnerabilities is crucial for conducting a risk analysis and determining the required risk level for a given function. To enable this process, the Vulnerability and Threat Modelling component provides the means to store and manage a knowledge base comprising of well-known vulnerabilities (i.e., functional requirement FR_RA_5). On the one hand the component supports the automated and periodic synchronization with publicly available repositories such as the National Vulnerability Database (NVD) [23]. These repositories provide additional information that enables the calculation of the impact of a vulnerability based on the Common Vulnerability Score System specification (CVSS) [26]. In addition, this component leverages other cybersecurity awareness initiatives that focus on the association of specific vulnerabilities with common attack patterns (CAPEC [1]), weakness (CWE [4]), and product catalogues (CPE [2]) that enable the seamless association of vulnerabilities with specific product identifiers.

In parallel, towards meeting the final functional requirement (i.e., FR_RA_12) related to the harmonization of the risk results of various methodologies, the Vulnerability and Threat Modelling component adopts the STRIDE modelling framework [7]. According to the specification, potential threats can be categorized into into six types: Spoofing, Tampering, Repudiation, Information Disclosure, Denial of Service, and Elevation of Privilege. By adopting the STRIDE model, it is possible to model threats in an interoperable fashion across risk assessment methodologies. Specifically, this classification is essential for TARA methodology in the context of the threat scenario identification, while for the CVSS-based methodology it is required for the calculation of the IRL value as mentioned in Subsection 7.2.2. Finally, to further extend the knowledge base of vulnerabilities, the component enables security administrators to include more vulnerabilities from other sources, including zero-day vulnerabilities identified in their CCAM infrastructure.

Damage Scenario Management Component

The definition of damage scenarios is pivotal in determining the risk associated with threat scenarios within the TARA methodology (refer to Section 7.2.1). The damage scenario management component contributes to the seventh functional requirement as it facilitates the fine-tuning of each identified damage scenario's impact, enabling security administrators to refine their risk assessment analysis. The precise definition of the four different types of impact (i.e., Financial, Operational, Safety, and Privacy impact) for each damage scenario allows the administrator to achieve a more accurate risk assessment outcome. Hence, this component supports the management of all the contributing factors of the impact of a damage scenario as well as the way these impact values are used to determine the overall impact that is used in the TARA methodology. For example, a security administrator may want to select the maximum impact that a damage scenario may have in any of the four aspects.

Attack Path Calculator

The identification of attack paths is a crucial (FR_RA_6 functional requirement) process for the realization of the TARA methodology. The attack path calculation component enables the threat scenario identification step in scope of the TARA methodology. This means that security administrators have the ability to specify the attack paths in the monitored infrastructure so as to allow the Threat Scenario Management component to parameterize the necessary attributes associated with the Attack Feasibility Rating as per the TARA methodology.

Similarly, the attack path calculator component is also critical for the enhancement of the risk overview in the context of the CVSS-based methodology through the calculation of the CRL value (Section 7.2.2). In scope of the OLISTIC artifact, it is possible to automate the attack path based on the vulnerability characteristics on neighbouring assets. Specifically, through the definition of a set of declarative rules, a security administrator defines the conditions under which an adversary can exploit a vulnerability on an asset that allows them to pivot to a neighbouring asset. These rules are then fed to the Drools engine [5], an expert system, designed for rule-based decision making and complex event processing.

Note: *The automated identification of attack paths with pre-defined rules poses significant limitations. It requires that the cascading effects are described beforehand by security administrators, thus, neglecting the necessity of capturing zero-day vulnerabilities. In scope of the attack path calculator, additional attack path identification approaches will be examined. The ultimate goal of this component is to contribute to the automation techniques applied to the TARA methodology that will reduce the workload required for the initial threat analysis. This is also aligned with the* FR_RA_11 *functional requirement.*

Attack Feasibility Recommendation Engine

The main scope of the Attack Feasibility Recommendation Engine is to facilitate the automation of the TARA methodology (FR_RA_11 functional requirement), through the use of the CVSS-based methodology. For each threat scenario, the CRL value - presented in Subsection 7.2.2 - can be used to express an initial recommendation for the attack feasibility rating. This can speed an initial round of the TARA methodology. Of course, security administrators will have the ability to adjust the recommendation of this component, as well as manually overwrite it by using the

attack feasibility rating attributes defined in the TARA methodology. The impact of this approach to the overall RTL calculations is evaluated in D3.3 [14].

Risk Quantification Engine

The risk quantification engine is responsible for consuming the threat analysis performed either automatically or manually by the security administrators and compute a risk assessment risk values for the monitored asset cartography. This component is designed in a modular fashion to allow the inclusion of diverse risk assessment methodologies. It supports the CVSS-based methodology for both the calculation of IRL and CRL values. In addition, it consumes the information provided by the TARA-related subcomponents to perform the risk analysis specified in ISO/SAE 21434 [19]. Having the results of multiple risk assessment methodologies, the risk quantification engine is also responsible for harmonizing the results in order to provide a transparent output to the maximum possible extent. However, this introduces a challenge that the same attack vectors for a given trust property are not taken into consideration in multiple methodologies as this may lead to an increased required trust level that may be difficult - or even impossible - to be satisfied by the in-vehicle components during runtime.

Furthermore, in scope of the FR_RA_7 functional requirement, the Risk Quantification Engine is in charge of managing the entire lifecycle of a risk assessment task, from initiation, to completion. In parallel, this component provides the various risk reports to be consumed by the OLISTIC Frontend as well as in scope of the RTL and ATL calculations.

Finally, apart from its modular design (i.e., supporting various risk assessment methodologies), the risk quantification engine provides the functionality of converging the risk results to the maximum possible extent (FR_RA_12 functional requirement). This provides a seamless integration between the CONNECT RA Engine and the trust calculations as security administrators may evaluate how different risk methodologies impact the RTL and ATL values. The details of the convergence implementation details are presented in D3.3 [14].

Mitigation Strategy Component

Two out of the eleven functional requirements of the CONNECT RA Engine are - namely, FR_RA_7 and FR_RA_8 - refer to the management of the various security controls and their impact on the risk level within the monitored topology. In the context of CONNECT, there are three major types of security controls:

- *Misbehaviour detection:* These controls are primarily focused on identifying and responding to anomalous or unauthorized behavior within the in-vehicle topology. Examples include the CONNECT MD component, intrusion detection systems (IDS), and security information and event management (SIEM) systems. These controls typically operate based on predefined rules or behavioral patterns and are designed to alert administrators or trigger automated responses when suspicious activity is detected.
- *Implemented security controls:* This category includes security measures that are in place and configured correctly but may not provide real-time evidence of their effectiveness during runtime. Examples include encryption protocols like MACsec, secure boot, and secure configurations.

- *Implemented and executed security controls (i.e., Runtime attestation controls):* These controls not only include security measures that are implemented but also provide ongoing evidence of their execution and effectiveness during runtime. Examples include runtime attestation protocols, integrity measurement frameworks, and control flow attestation. These controls provide additional levels of assurance, allowing administrators to verify that security measures are actively enforced and responding as expected to potential threats.

The Mitigation Strategy Component supports the modeling of all types of controls. It also allows security administrators to signal the effectiveness of each security control by configuring the various parameters that contribute to the overall risk level for an asset or a trust relationship. By combining multiple security controls for various assets, security administrators can define specific profiles, namely Mitigation Strategies. The component enables the management of the various mitigation strategies and the evaluation of their impact on the overall risk. The mitigation strategy risk results are essential for initializing the ATL calculations during design time. As presented in Subsection 7.4.1, this risk array can be pre-configured in the TAF components during design phase. At runtime, this information can be used by the ATL calculation to select the suitable risk value, depending on the available attestation evidence reported in the Trust Source Manager.

7.4.4 CONNECT RA Framework

This section provides the sequence diagram showcasing the workflow of a risk assessment process in the context of CONNECT. This workflow describes all the interactions between the internal components of the CONNECT RA Framework and highlights the points where the CVSS-based methodology could offer automation capabilities to the TARA methodology. The aim is to specify how the CONNECT RA Engine reacts to a Request For Risk Assessment (RFRA) message up to the computation of the final TARA-related risk results. The integration of the component within the design phase of the Trust Assessment Management is presented in Subsection 7.4.1.

Figure 7.4 presents the sequence of actions starting from the Request For Risk Assessment (RFRA). This request can be triggered by the security administrator through the OLISTIC Frontend, or through the API interface. This request can also be a result of a security incident related to the asset topology (i.e., a new vulnerability added to the asset topology triggers the initiation of a new risk assessment). It is assumed that the necessary threat analysis has already been performed, including the asset topology, the vulnerability analysis, and the rest of the TARA configuration. Given that the CONNECT RA Engine has an asset graph corresponding to each trust model identified by the security administrators and/or OEMs, the RFRA payload should specify the specific graph for which the risk assessment is to be executed.

The Risk Quantification Engine uses the RFRA payload to fetch the necessary asset graph information from the Asset Modeling and Visualization Engine. Subsequently, for each of the corresponding assets, the associated vulnerability information is fetched through the Vulnerability and Threat Modeling component, including the respective CVSS and threat probability information. The Risk Quantification Engine invokes the CVSS-based implementation to compute the impact of each vulnerability and, eventually, the individual risk level for each $(Asset, Vulnerability, Threat)$ tuple according to Subsection 7.2.2.

Next, the Attack Path Calculator is invoked, which uses the CVSS-based methodology risk results and associated information to identify the Attack Paths in an automated manner. These

attack paths may also include information manually added by the security administrator. These attack path results are forwarded to the Attack Feasibility Recommendation Engine that uses the CRL, CVSS-based, risk values to provide a recommendation to the Attack Feasibility rating for the TARA methodology. This information is returned to the Risk Quantification Engine which is able to derive the TARA threat scenario information. In fact, each attack path may result in multiple threat scenarios, depending on the damage scenarios that impact the target asset. Based on the damage scenario impact information that is pre-defined, and the Attack Feasibility recommendation it is possible to construct the TARA risk level for each identified threat scenario, thus forming the risk scenarios. The risk scenarios are now available for consumption on the OLISTIC Frontend as well as the RTL calculations within the Trust Management Framework.

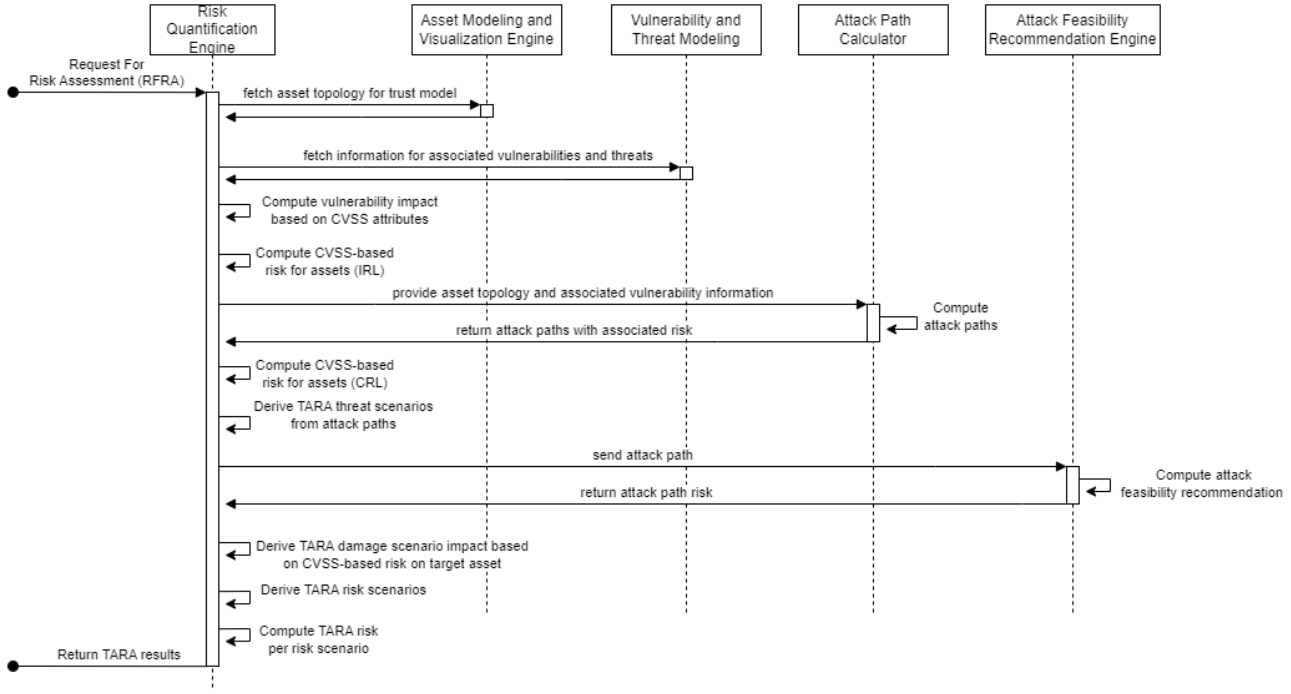


Figure 7.4: Risk Assessment sequence diagram

7.5 Risk Assessment Engine API Specification

This subsection presents the API specification of the CONNECT RA Engine. Following the architecture schema in Figure 7.2, the set of APIs can be depicted in two major categories: the internal API which is related to all the communication among components within the CONNECT Cloud Infrastructure. Part of the internal communication is the one that is consumed by the TAM component in the context of the RTL calculations. In parallel, there are the external API endpoints referring to the invocations made by entities that can be external to the CONNECT Cloud infrastructure or the data that are produced are consumed by such external entities. This is the case for all the interfaces pertaining to the threat analysis by an OEM, the communication for the ATL calculations as well as the reception of indication of risks (e.g., failed attestation evidence from the in-vehicle devices). It is possible that API endpoints characterized as external can also be invoked internally (e.g., through the OLISTIC Frontend interfaces); however, to avoid repeatability, endpoints that have been mentioned in Subsection 7.5 are not repeated in the first table.

Internal API Specification

All the internal interactions among the subcomponents of the CONNECT RA Engine are performed through specific REST interfaces as presented in Table. Apart from these, a KAFKA communication channel is established to send asynchronously the risk assessment results to the Trust Assessment Manager for the realization of the RTL calculations.

Name: VULNERABILITY_THREAT_INTERFACE			
Description	Interface offered by the Vulnerability and Threat Modeling Component for managing threats, vulnerabilities. This is essential for enabling the threat analysis prior to any risk assessment task.		
Component providing the interface	Vulnerability and Threat modeling component		
Consumer components or External Entities	Risk Quantification Engine, OLISTIC Backend, OLISTIC Frontend, External OEM integration activities		
Type of interface	REST		
Input /Output Data	**Methods or endpoints of the interface**	**Parameters of the method**	**Return Object or Values of the method**
	GET /api/v1/vulnerabilities /{id}	Vulnerability Identifier	Vulnerability Information
	POST /api/v1/vulnerabilities /{id}	Vulnerability Information	Vulnerability Identifier and name
	DELETE /api/v1/vulnerabilities /{id}	Vulnerability Identifier	HTTP OK
	GET /api/v1/threats /{id}	Threat Id	Threat information
	POST /api/v1/threats /{id}	Threat Information	Threat Identifier and name
	DELETE /api/v1/threats /{id}	Threat Identifier	HTTP OK

Table 7.4: Vulnerability and Threat Modeling component API

Name: ASSET_MODELLING_INTERFACE			
Description	Interface offered by the Asset modeling and visualization component to enable security administrators to manage assets and asset topologies. This will enable the execution of fine grained risk analysis in pre-defined asset topologies corresponding to different trust models.		
Component providing the interface	Asset modeling and visualization component		
Consumer components or External Entities	Risk Quantification Engine, OLISTIC Backend, OLISTIC Frontend, External OEM integration activities		
Type of interface	REST		
Input/Output Data	**Methods or endpoints of the interface**	**Parameters of the method**	**Return Object or Values of the method**
	GET /api/v1/assets /{id}	Asset Identifier	Asset information
	POST /api/v1/assets /{id}	Asset Information	Asset Identifier and name
	DELETE /api/v1/assets /{id}	Asset Identifier	HTTP OK
	GET /api/v1/processes /{id}	Process Id	Process information
	POST /api/v1/processes /{id}	Process Information	Process Identifier and name
	DELETE /api/v1/processes /{id}	Process Identifier	HTTP OK
	GET /api/v1/processes /{id}/visualize	Process Id	Asset topology information

Table 7.5: Asset Modeling and Visualization component API

Name: ATTACK_PATH_CALCULATOR_INTERFACE			
Description	Interface offered by the Attack Path Calculation component to manage attack path assessments as well as the update of the already identified attack paths with new ones by the security administrator.		
Component providing the interface	Attack Path Calculator		
Consumer components or External Entities	Risk Quantification Engine, OLISTIC Backend, OLISTIC Frontend, External OEM integration activities		
Type of interface	REST		
Input/Output Data	**Methods or endpoints of the interface**	**Parameters of the method**	**Return Object or Values of the method**
	GET /api/v1/attack-path-assessment /{id}	Attack Path Assessment Identifier	Attack path assessment information
	POST /api/v1/attack-path-assessment /{id}	Attack path assessment information	Attack path assessment Identifier and name
	DELETE /api/v1/attack-path-assessment /{id}	Attack path assessment Identifier	HTTP OK
	POST /api/v1/attack-path-assessment /{id}/execute	Attack path assessment Id	HTTP OK

	POST /api/v1/attack-path-assessment /{id}/visualize	Attack path assessment Id	Attack paths in the format of a graph.

Table 7.6: Attack path calculator component API

Name: RISK_QUANTIFICATION_ENGINE_INTERFACE			
Description	Interface offered by the risk quantification engine that coordinates risk assessment tasks from the submission to their completion		
Component providing the interface	Mitigation Strategies component		
Consumer components or External Entities	Risk Quantification Engine, OLISTIC Backend, OLISTIC Frontend, External OEM integration activities, TAM		
Type of interface	REST		
Input/Output Data	**Methods or endpoints of the interface**	**Parameters of the method**	**Return Object or Values of the method**
	GET /api/v1/risk-assessments /{id}	Risk Assessment Identifier	Risk Assessment information
	POST /api/v1/risk-assessments /{id}	Risk Assessment Information	Risk Assessment Identifier and name
	DELETE /api/v1/risk-assessments /{id}	Risk Assessment Identifier	HTTP OK

Table 7.7: Risk Assessment strategies component API

Name: MITIGATION_STRATEGIES_INTERFACE			
Description	Interface offered by the mitigation strategies component to enable security administrators to manage the security controls that are associated with specific assets.		
Component providing the interface	Mitigation Strategies component		
Consumer components or External Entities	Risk Quantification Engine, OLISTIC Backend, OLISTIC Frontend, External OEM integration activities, TAM		
Type of interface	REST		
Input/Output Data	**Methods or endpoints of the interface**	**Parameters of the method**	**Return Object or Values of the method**
	GET /api/v1/controls /{id}	Control Identifier	Control information
	POST /api/v1/controls /{id}	Control Information	Control Identifier and name
	DELETE /api/v1/controls /{id}	Control Identifier	HTTP OK

Table 7.8: Mitigation Strategies component API

Interface description with the CONNECT framework (External interfaces)

Source	Target	ID	Description	Content	Technology
OEM	RA	RA_EX_1	Detection of a new vulnerability (e.g., through the inspection of failed attestation evidence)	Vulnerability ID, Asset ID and Vulnerability Information (CVSS attributes)	REST
OEM	RA	RA_EX_2	Update of the asset topology in scope of the OEM threat analysis	Asset ID, Asset topology ID, Asset attributes (e.g., CPE) and Asset Relationships	REST
OEM	RA	RA_EX_3	Manage (Create, Update, Read, Delete) vulnerabilities and threats in the CONNECT RA knowledge base	Vulnerability Information (including CVSS attributes) and/or Threat information (including Threat probability)	REST
OEM	RA	RA_EX_4	Manage (Create, Read, Update, Delete) Damage Scenarios	Damage Scenario information, including impact values and associated asset IDs	REST
OEM	RA	RA_EX_5	Manage (Create, Read, Update, Delete) Threat Scenarios	Attack path asset ID list, with exploited vulnerabilities and Attack Path Feasibility information	REST
OEM	RA	RA_EX_6	Security controls	Security control ID and/or security control information including associated vulnerability and threat IDS that it mitigates	REST

OEM	RA	RA_EX_7	Manage (Create, Read, Update, Delete) mitigation strategies	Mitigation Strategy ID, and/or Control ID list that is part of a mitigation strategy	REST
TAM(ATL)	RA	RA_EX_8	Evaluation of effectiveness of security controls	Request for Risk Assessment with the asset topology ID and a list of mitigation strategy IDs to be evaluated	KAFKA

Table 7.9: Overview of external interfaces

7.6 CONNECT RA Implementation Roadmap

This section presents the CONNECT Risk Assessment Engine implementation roadmap. By the final version, all the functional specifications presented in Section 7.3 are going to be incorporated. The roadmap consists of three major milestones as highlighted in Figure 7.5. In the first milestone (M18), the CONNECT RA Engine is able to provide the framework for enabling the threat analysis, conducting CVSS-based risk assessment tasks, and exposing results to the users. Subsequently, an updated version (M21) incorporates the TARA methodology including all the necessary threat analysis steps to configure the framework (e.g., damage scenarios, attack feasibility rating parameters). This version is planned to be used for the first evaluation of the use cases (M24) where the CONNECT RA Engine and the RTL calculations are integrated. By the second and final version (M27), the CONNECT RA Engine supports the automated attack path calculation functionality along with harmonization mechanisms for the two risk assessment methodologies and the integration with the ATL calculations. The implementation details are reported in D3.3 [14] and D6.3 [15]. This final version of the risk assessment framework is used for the final use cases evaluation (M30) to evaluate the complete set of CONNECT RA Engine features (i.e., as depicted in Subsection 7.3) as well as the RTL and ATL calculations.

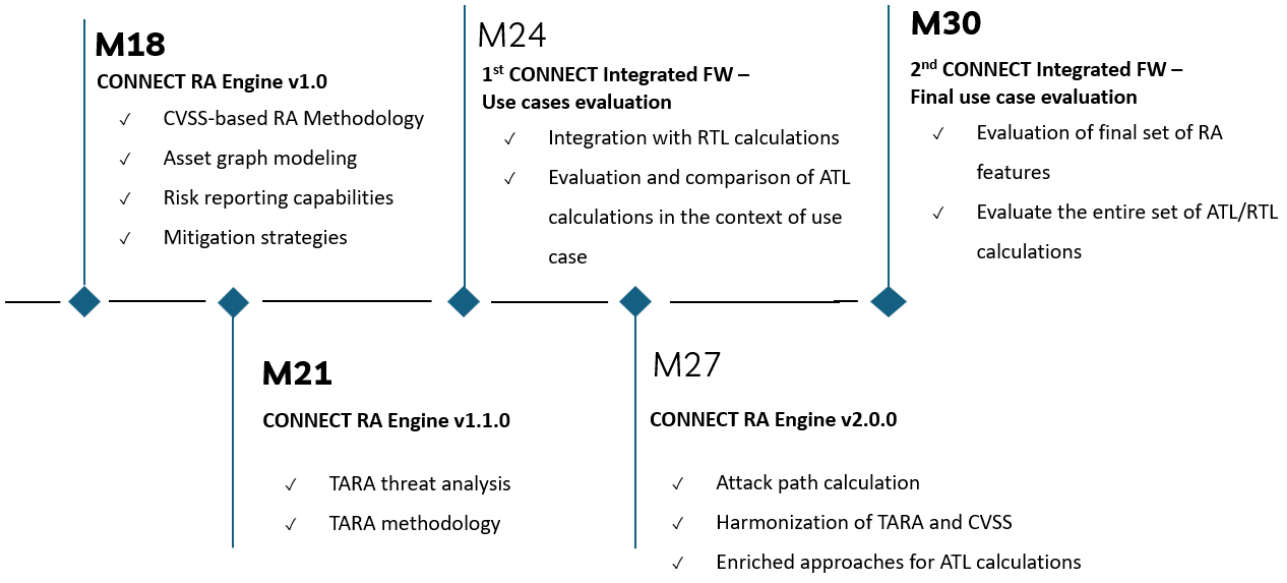


Figure 7.5: CONNECT RA Engine implementation roadmap

7.7 Running Example on In-Vehicle Function Risk Assessment

This chapter refers to Chapter and the example described in chapter 3 to illustrate the risk assessment methodology considered for CONNECT. Furthermore, this use-case is used as basis for the examples shown in the chapters 9 and 8. The first section shows a simplified TARA for the architecture shown in Figure 3.1. The second section exemplifies the CVSS-based risk assessment methodology discussed in 7.2.2.

7.7.1 TARA applied to the running example

In this section is defined and given more detailed information about the running example, considering cybersecurity aspects and following the steps standardised in [19]. The tables in this section are populated based on possible threats or damage scenarios to be used as examples, not necessarily expressing the reality.

Item definition

The item function is to receive GNSS data and make it accessible on the vehicle's computer.

The item is composed of GNSS data and GNSS ECU (represented by GNSS symbol in the figure), Zonal Controller 1 and Vehicle Computer. The communication between them is considered unprotected. The operational environment is composed of a Lidar and Zonal Controller 2, since they have direct connection with the item.

Asset identification and damage scenario

The Table 7.10 shows a list of damage scenarios which may compromise cybersecurity properties of the use case.

ID	Asset	Damage scenario	Security property
DS.1	GNSS data	The vehicle location can be accessed by an attacker	Confidentiality
DS.2	GNSS data	The GNSS data can be adulterated by an attacker	Integrity
DS.3	Vehicle computer	The firmware on the vehicle computer can be changed by an attacker	Integrity
DS.4	Zonal controller	The firmware on the zonal controller can be changed by an attacker	Integrity
DS.5	ECU	The data stored on the ECU can be accessed by an attacker	Confidentiality
DS.6	GNSS data	There's no GNSS signal and the vehicle cannot be located	Availability
DS.7	Vehicle computer	Application is compromised by an attacker	Integrity
DS.8	Vehicle computer	Firmware compromised by an attacker	Integrity
DS.9	System	An attacker can impersonate an entity to perform restrictive actions...	Authorisation
...	...	...	...

Table 7.10: Damage scenario and asset identification

Threat scenario identification and attack feasibility

The damage scenarios can be realised by threats. The Table 7.11 shows a list of threats that can be performed against the use case scenario.

ID	Damage scenario	Threat scenario	Attack feasibility
TS.1	DS.1, DS.5	Information Disclosure on ECU	Medium
TS.2	DS.1	Information Disclosure on Zonal controller	Low
TS.3	DS.1	Information Disclosure on Vehicle computer	Very Low
TS.4	DS.2	Tampering on GNSS data	High
TS.5	DS.2, DS.3	Tampering on Vehicle computer	Very Low
TS.6	DS.2, DS.4	Tampering on Zonal controller	Low
TS.7	DS.6	Physical interference	Medium
TS.8	DS.7	Compromised application	Medium
TS.9	DS.2	Man-in-the-Middle external communication	High
TS.10	DS.3, DS.4, DS.8	Compromised network firmware	High
TS.11	DS.2	Man-in-the-Middle internal communication	Low
TS.12	DS.9	Impersonation attack	High
TS.13	DS.5	Physical attack on RAM	Low
...	...	...	...

Table 7.11: Threat scenario identification and attack feasibility

Impact rating

The impact rating for the categories safety (s), financial (F), operational (O) and privacy (P) is shown in Table 7.12 for each damage scenario.

Damage scenario	S	F	O	P
DS.1	Negligible	Negligible	Negligible	Moderate
DS.2	Moderate	Negligible	Major	Moderate
DS.3	Severe	Negligible	Severe	Negligible
DS.4	Moderate	Moderate	Severe	Negligible
DS.5	Negligible	Negligible	Negligible	Moderate
DS.6	Negligible	Negligible	Major	Negligible
DS.7	Severe	Negligible	Severe	Negligible
DS.8	Negligible	Negligible	Severe	Negligible
DS.9	Severe	Negligible	Severe	Negligible
...	...	...	...	...

Table 7.12: Impact rating

Risk level determination

To translate impact rating and attack feasibility rating into risk level, with a numeric value, we are using the values in the Tables 7.13 and 7.14, also used for the example shown in the ISO/SAE 21434 [19].

Impact Rating	Numerical value I for impact rating
Negligible	0
Moderate	1
Major	1.5
Severe	2

Table 7.13: Translation of impact rating to a numerical value [19]

Attack feasibility	Numerical value F for attack feasibility
Very Low	0
Low	1
Medium	1.5
High	2

Table 7.14: Translation of attack feasibility to a numerical value [19]

For the risk level (R) calculation, we are using the equation 7.1, from the example in [19].

$$R = 1 + I \times F \tag{7.1}$$

The translated risk level is shown in the table 7.15.

	Very Low	Low	Medium	High
Severe	1	3	4	5
Major	1	2.5	3.25	4
Moderate	1	2	2.5	3
Negligible	1	1	1	1

Table 7.15: Risk level calculation considering the equation 7.1 [19]

The Table 7.16 shows the risk level for each threat scenario and damage scenario listed in the tables 7.11 and 7.12.

ID	Threat scenario	Damage scenario	S	F	O	P
R.1	TS.1	DS.1	1	1	1	*2.5*
R.2	TS.1	DS.5	1	1	1	*2.5*
R.3	TS.2	DS.1	1	1	1	*2*
R.4	TS.3	DS.1	1	1	1	1
R.5	TS.4	DS.2	3	1	*4*	3
R.6	TS.5	DS.2	1	1	1	1
R.7	TS.5	DS.3	1	1	1	1
R.8	TS.6	DS.2	2	1	*2.5*	2
R.9	TS.6	DS.4	2	2	*3*	1
R.10	TS.7	DS.6	1	1	*4*	1
R.11	TS.8	DS.7	1	1	*4*	1
R.12	TS.9	DS.2	1	1	*4*	1
R.13	TS.10	DS.3	1	1	*5*	1
R.14	TS.10	DS.4	1	1	*5*	1
R.15	TS.10	DS.8	1	1	*5*	1
R.16	TS.11	DS.2	1	1	*3*	1
R.17	TS.12	DS.9	1	1	*5*	1
R.18	TS.13	DS.5	1	1	*2*	1

Table 7.16: Risk level

Based on TARA, a possible next step would be to define cybersecurity controls to mitigate the listed threats, if possible, but this is not part of this example. The Chapter 8 covers a little about cybersecurity controls and its uses in the trust opinion calculation.

7.7.2 CVSS-based methodology calculations

Table 7.17 presents an example set of vulnerabilities associated with each of the cyber-physical assets. Each vulnerability is uniquely identified with a CVE identifier as well as with the CVSS characteristics that enable the computation of the vulnerability impact. In this chain of assets, it is evident that the most impactful vulnerability is the one in the vehicle computer that allows an adversary to perform arbitrary code remotely. To perform the CVSS-based methodology presented in subsection 7.2.2 it is required that each vulnerability is associated with a specific threat. Hence, in the context of our threat analysis within this simple example, we also provide the most prominent threat that an adversary may achieve in the in-vehicle topology. The threat classification follows the STRIDE model. The last piece of information required is the subjective parameter of the threat probability, signaling the likelihood of occurrence of each threat in the monitored topology. The result of the risk value for each asset is depicted in the IRL_1 line. The quantitative score is the product of the vulnerability impact and the threat probability. In scope of risk mitigation, a user may select to enforce a control flow integrity protocol (see [11]) in the Vehicle Computer component to tackle any memory-related vulnerability. In the risk calculations, this can be reflected through the updated threat probability value for the Tampering threat by setting it from ”Very High” to ”Very Low”. As shown in the IRL_2 calculations, the IRL value for the Vehicle Computer component has been significantly reduced by 90%.

Asset	GNSS data	ECU	Zonal controller	Vehicle computer
Vulnerability	-	CVE-2023-6246	CVE-2021-46145	CVE-2023-40547
Vuln. Impact	-	7.8	5.3	8.3
Threat	-	Elevation of Privilege	Authentication Bypass	Tampering
Threat Probability	-	High	Medium	Very High
IRL_1	-	$0.8 \times 0.78 = 0.62$	$0.5 \times 0.53 = 0.27$	$1.0 \times 0.83 =$ **0.83**
IRL_2	-	$0.8 \times 0.78 = 0.62$	$0.5 \times 0.53 = 0.27$	$0.1 \times 0.83 =$ **0.08**

Table 7.17: CVSS-based risks and associated information per asset.

In this section we introduced the running example including the component diagram and data flows. In addition, we carried out the threat analysis required both for the TARA and CVSS-based methodologies presented in this chapter. This analysis led to the calculation of the respective risk values for the involved assets, including the provision of security controls in the calculations. In the following chapters the risk quantification results are used to provide the necessary RTL values for each trust entity and eventually for the overall CCAM function (9) presented in this example. In parallel, with respect to the ATL calculations (8) the CONNECT RA Engine provides the risk quantification per identified mitigation strategy as well as an evaluation of the impact (i.e., effectiveness) of the enforced security controls to the asset graph. This information enables the ATL calculations at runtime, to collect the appropriate set of trustworthiness evidence to validate the presence of specific security controls.

Chapter 8

Evidence-based Trust Opinion Calculation

The Trust Source Manager (TSM) is a component of the TAF that is responsible for managing the single trust sources specified in the trust model. As already described in Chapter [**?**], there exists a trust source quantifier instance (TSQ-I) for each trust source managed by the TSM. The TSQ-I is responsible for processing the evidence received from the trust source and calculating an atomic trust opinion based on that. Depending on the trust source and the type of evidence provided by this trust source, the TSQ-I and the approach to calculate the trust opinion are very different. The following part of this chapter describes approaches for different trust sources on how the corresponding TSQ-I can calculate a trust opinion.

8.1 Trust Assessment based on Misbehaviour Detection

A misbehavior detection system can be used as a trust source for trust relationships from a node, e.g. an ECU, on a data item, e.g. position data. In the context of the CONNECT project, the misbehavior detection system is used as a trust source to create a trust opinion not only on a single data item but on a whole observation. A node gets observations by receiving CAMs or CPMs from other nodes. A CAM contains only one observation about the sender. A CPM contains several observations about the sender and its environment. An observation describes in this context kinematic data. The kinematic data contains several data items such as position data and speed data and characterizes in this way either the message transmitter or other vehicles in the environment. Based on the output of the misbehavior detection system, a trust opinion is created on the whole observation and thus on several data items. Misbehavior detection is here a mechanism to detect anomalous behavior by analysing the semantics of the transmitted messages. It contains several misbehavior detectors which check the plausibility and the consistency of the kinematic data to detect potential misbehavior. To create a trust opinion on the kinematic data, the misbehavior detection system running on the node which is assessing the trust opinion, provides the output of the single misbehavior detectors to the TSM. Alternatively, the TSM receives a misbehavior report with the output of the misbehavior detection system running in another node, such as another vehicle.

The task of the TSQ-I for misbehavior detection systems is to create based on the output of the misbehavior detection system a trust opinion on an observation. A misbehavior detection system contains several misbehavior detectors and provides the output of each misbehavior detector to the TSM. These misbehavior detectors are described in the following.

8.1.1 Misbehavior Detectors

Each detector of the misbehavior detection system provides an output if it detected malicious behavior. For these misbehavior detectors several activation patterns are possible. The activation patterns describe the output provided by each misbehavior detector. A possible activation pattern is a binary pattern. With the binary activation pattern, the detector can only provide zero or one as an output. If the output is zero, this means that the misbehavior detector has not detected anything. If the output is one, the misbehavior detector has detected malicious activities. In addition to the binary activation pattern there exist further activation patterns, such as a floating pattern. In the floating activation pattern, each detector provides a float value that describes how certain the detector is that it has detected malicious activities. However, in this project we focus on the binary activation pattern, as this pattern is supported by the misbehavior detection system used in the simulation environment within the CONNECT project.

There exist several misbehavior detectors that can be used in a misbehavior detection system. We focus on misbehavior detectors that are relevant to detect a compromised position and velocity of an observation. The reason for this is that the misbehavior detection system is used in the intersection movement assist use-case. In this use-case only the position and velocity are relevant. However, the provided approach for the calculation of the trust opinion could be adjusted to an arbitrary number of attributes. In the following, we provide an overview of the misbehavior detectors that are used to create a trust opinion for the position and velocity [22].

- Range Plausibility: Check if the position of the sending node is inside of the maximum perception range of the ego node (predefined threshold value).
- Position plausibility: Check if the position of the sending node is in a plausible location. It is for example checked, if the position is on a road or if no overlaps exist with physical obstacles.
- Speed plausibility: Check if the speed provided by the sending node is lower than an specified threshold value.
- Position consistency: Check if two CAMs from the same sender have a plausible distance (less than a predefined threshold).
- Speed consistency: Check if two CAMs from a same sender have a plausible acceleration and deceleration (less than a predefined threshold).
- Position speed consistency: Check if the distance of two CAMs from the same node is consistent with the advertised speed.
- Position heading consistency: Check if the positions in two CAMs from the same node correspond to the heading provided by that node.
- Intersection check: Check if the positions of two CAMs from two different nodes are overlapping.
- Kalman filter tracking: Check if the information provided by a node is in a plausible range of the predicted values of a Kalman filter.

8.1.2 Architecture of Trust Assessment Process

Figure 8.1 shows the architecture used to calculate the trust opinion based on the output of the single misbehavior detectors. The left part of the figure represents the misbehavior detectors used to create the trust opinion for the position data. The right part of the figure represents the misbehavior detectors used to create the trust opinion for speed data. Based on these two trust opinions and a fusion operator, the final trust opinion on the observation is calculated. There are several fusion operators provided by the subjective logic framework. Which operator to use here, will be specified at design-time in the TSQ-T which is assigned to misbehavior detection. A possible fusion operator could be the weighted fusion operator. The calculated trust opinion is finally stored in the trust model.

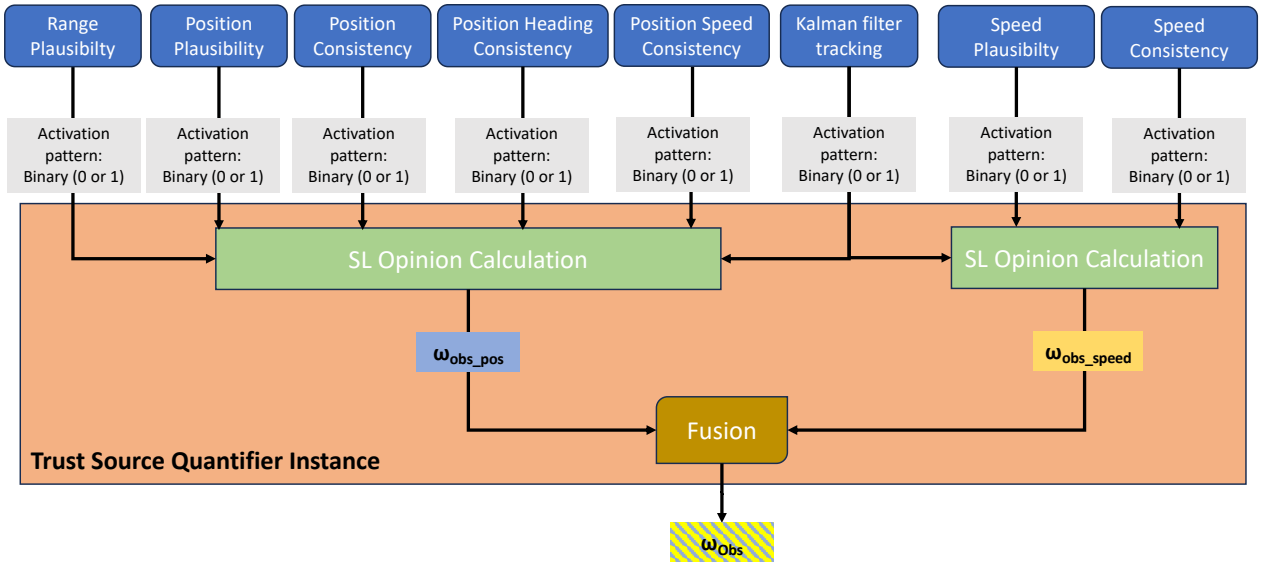


Figure 8.1: Single misbehavior detectors to assess the trustworthiness of an observation (position and speed).

Based on the output of the single misbehavior detectors, in each green box the corresponding trust opinion is calculated. The first trust opinion represents the trustworthiness of the position and the second trust opinion represents the trustworthiness of the speed. The single SL calculation components and the fusion component are part of the TSQ-T for misbehavior detection systems. How the single trust opinions for the position and speed are calculated, is described in the following part of this section.

8.1.3 Approach for Trust Opinion Calculation

To calculate the trust opinion for the position and the speed, several attributes are necessary, which have to be specified at design-time. These attributes are specified as metadata in the trust model template. For each data item it is specified which misbehavior detectors are foreseen and should be used to analyze this data item. Furthermore, for each data item it is specified how much uncertainty this detector has in its decision. As we are using binary activation patterns, this uncertainty can not dynamically be provided for each detector. Therefore, this has to be specified individually for each detector, e.g. based on knowledge or past experience with this

detector. In addition to that, also weights are specified for each detector. These weights describe the importance of the corresponding detector that the data item is compromised. Some detectors, such as the speed plausibility detector, perform rather simple checks and are therefore not as accurate in detecting malicious behavior as other detectors, such as the position heading consistency detector, which do more enhanced checks and can therefore detect attacks more accurately. This should be taken into account in the weights and eventually in the quantification of the trust opinion.

Thus, there are several inputs necessary for the trust assessment based on misbehavior detection. In the first step, we use a rather simple approach and set 0.2 as the uncertainty for each detector. Furthermore, we set the weight for each detector to the same value. As there are in total six foreseen detectors for the position, each position detector has a weight of $1/6 = 0.167$. For the speed there are in total three foreseen detectors. Thus, each detector has a weight of $1/3 = 0.33$.

The **SL Opinion Calculation** component shown in Figure 8.1 receives the output of the single misbehavior detectors. Based on this output the left SL Opinion Calculation component is calculating the trust opinion of the position. The right SL Opinion Calculation component is calculating the trust opinion of the speed. For the calculation of the trust opinion, the weights specified for each detector are used. In the first step, the weight of the corresponding detector is used and multiplied by $1 - ufactor_{mbd\_x}$. As already mentioned above, the uncertainty-factor $ufactor_{mbd\_x}$ was set to 0.2 for each detector. Based on the weight and this uncertainty factor, the delta value can be calculated for each detector. The delta value describes how much the believe, disbelief and uncertainty are adjusted depending on whether the single misbehavior detector has detected something or not. This is done in that the weight is multiplied by $1-ufactor_{mbd\_x}$ and is described in the following equation:

$$delta_{mbd\_x} = weight_{mbd\_x} \times (1 - ufactor_{mbd\_x}) \tag{8.1}$$

To calculate the corresponding trust opinion, an initial trust opinion is used, which has maximum uncertainty ($\omega_{init} = (0, 0, 1)$), since there is no knowledge about the observation in the initial state. Depending on if the corresponding misbehavior detector detected something or not, either the belief will be increase by the $delta_{mbd\_x}$ value or the disbelief will be increase by the $delta_{mbd\_x}$ value. Independently of positive or negative evidence, the uncertainty will be decreased by $delta_{mbd\_x}$. This will be done for each individual misbehavior detector. In this way, the trust opinion for the position and the speed is calculated.

8.1.4 Example of Trust Opinion Calculation

A concrete example for the described approach is provided in the following for the position attribute. For the example we assume, that the position consistency detector detected malicious behavior. The other five remaining detectors for the position did not detect malicious behavior. The calculation of the trust opinion is shown in the following:

- Range Plausibility: Positive evidence (no detection)
  $\omega_{init} = (0, 0, 1) \Rightarrow \omega_1 = (0 + 0.167, 0, 1 - 0.167) = (0.167, 0, 0.833)$
- Position Plausibility: Positive evidence (no detection)
  $\omega_1 = (0.167, 0, 0.833) \Rightarrow \omega_2 = (0.167 + 0.167, 0, 0.833 - 0.167) = (0.334, 0, 0.667)$

- Position Consistency: Negative evidence (detection)
  $\omega_2 = (0.334, 0, 0.667) \Rightarrow \omega_3 = (0.334, 0 + 0.167, 0.667 - 0.167) = (0.334, 0.167, 0.5)$
- Position Heading Consistency: Positive evidence (no detection)
  $\omega_3 = (0.334, 0.167, 0.5) \Rightarrow \omega_4 = (0.334 + 0.167, 0.167, 0.5 - 0.167) = (0.5, 0.167, 0.333)$
- Position Speed Consistency: Positive evidence (no detection)
  $\omega_4 = (0.5, 0.167, 0.333) \Rightarrow \omega_5 = (0.5 + 0.167, 0.167, 0.333 - 0.167) = (0.667, 0.167, 0.167)$
- Kalman Filter Tracking: Positive evidence (no detection)
  $\omega_5 = (0.667, 0.167, 0.167) \Rightarrow \omega_6 = (0.667 + 0.167, 0.167, 0.167 - 0.167) = (0.834, 0.167, 0)$

The trust opinion $\omega_6 = (0.834, 0.167, 0)$ is the calculated trust opinion for the position based on the output provided by the misbehavior detectors. An overview of the approach for calculating the trust opinion for the position is shown in Figure 8.2. The calculation of the trust opinion for the speed is basically the same and is therefore not illustrated here again. The trust opinions for the position and the speed are fused together resulting in a trust opinion for the whole observation.

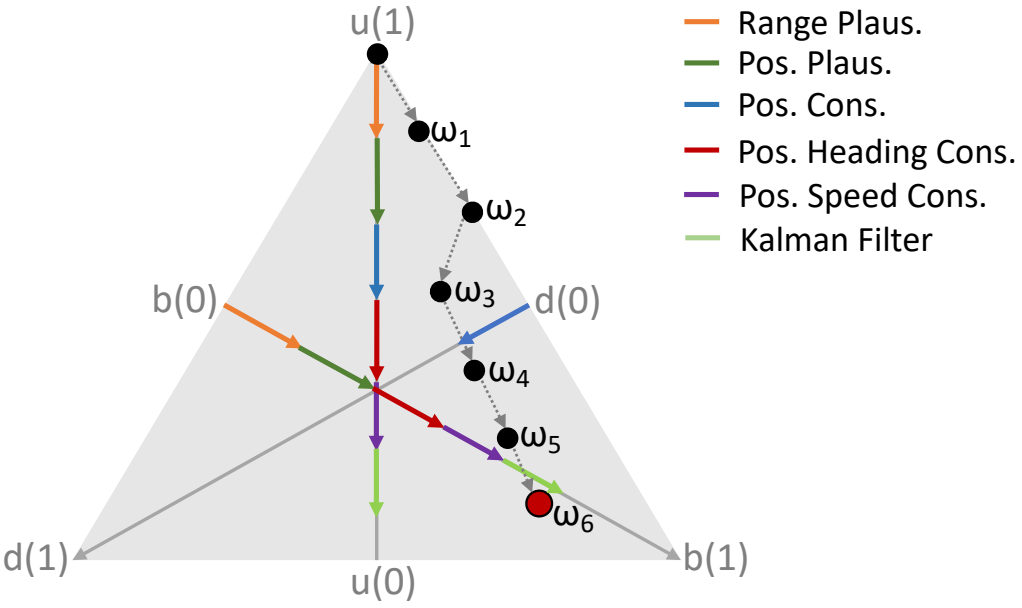


Figure 8.2: Calculation of a trust opinion based on the output of misbehavior detection.

8.2 Trust Assessment based on Security Controls

A further trust source is knowledge about the existence of security controls in a system to protect the integrity of data. We are focusing in this section on security controls in in-vehicle networks and assess the trustworthiness of ECUs based on knowledge of implemented security controls in that ECU. In this context, security controls could be, e.g., the integrity protection of data on a vehicle bus with MACsec. Depending on the security controls implemented in the ECU, there is a different risk of the ECU being compromised or the data provided by this ECU being compromised. Therefore, security controls are a mandatory source to assess the trustworthiness of an ECU. Knowledge about the applied security controls can come from Threat Analysis and Risk Assessments (TARA). Details of the TARA process were already described in Section 7.2.1. In this

section, it is described how the output provided by a TARA and knowledge about implemented security controls derived at run-time can be used to assess trustworthiness of an ECU.

As already mentioned above, the approach described in this section is limited to in-vehicle networks. How such an approach could be used to assess the trustworthiness of an entity outside of in-vehicle networks, e.g. other vehicles or RSUs, will be investigated in the further progress of the CONNECT project.

8.2.1 Running example

To demonstrate the calculation of the trust assessment approach, we are using a running example. The running example was already introduced beforehand in Section 3. Here, the GNSS-ECU provides position data to the vehicle computer. The corresponding Trust Model of this use-case is shown in Figure 8.3. It contains two nodes and one data item. Furthermore, the Trust Model has three trust relationships in the trust chain (ω_{ZC1}^{VC}, ω_{GNSS}^{ZC1}, and ω_{Pos}^{GNSS}). In our running example we focus on how the vehicle computer (VC) can form a trust opinion on the zonal controller 1 (ZC1) by using knowledge about security controls implemented in ZC1. This is represented in the Trust Model with the trust opinion ω_{ZC1}^{VC}. The same approach could be used for the other trust relationships between ECU nodes in the Trust Model. However, for trust relationships involving data items, other sources of evidence are necessary.

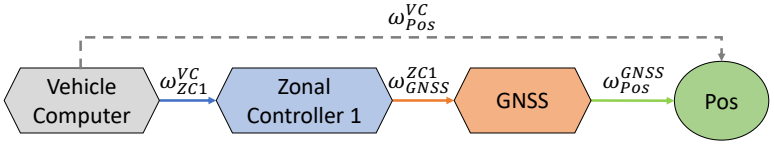


Figure 8.3: Trust Model derived from the in-vehicle network.

To assess the trustworthiness from the VC to the ZC1, the VC analyzes design-time and run-time information about the security controls in ZC1 to calculate the trust opinion ω_{ZC1}^{VC}. This opinion refers to the proposition that the integrity of the position forwarded by ZC1 was not compromised in ZC1. The final goal is to create the opinion ω_{Pos}^{VC} that the VC has on the position even though the VC does not observe the position directly (i.e., it does not have a direct relationship with the position). The single steps to calculate the trust opinion ω_{ZC1}^{VC} based on the used security controls are described in the following subsection.

8.2.2 Approach for Trust Opinion Calculation

Figure 8.4 shows an overview of the single steps conducted at design-time and at run-time to create a trust opinion for another ECU based on the implemented security controls in that ECU. In this approach the system knowledge is used at design-time to determine the risk levels in the ECU. Based on these risk levels and the evidence collected at run-time about implemented controls, the trust opinion for the ECU is calculated. The single steps of this approach are described in the following.

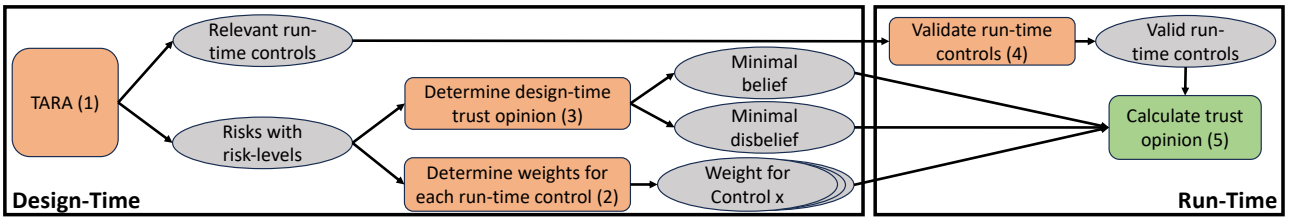


Figure 8.4: Overview of the trust assessment approach based on the output of the TARA

Security Control and Risk Level Determination (Step 1)

In the first step of the approach a TARA is conducted at design-time for the ECU for which the trust opinion is determined, ZC1 in the introduced running example. All components within ZC1 that process the position and transmit it to the VC are relevant assets analyzed by the TARA. The TARA process was already described in Section 7.2.1. As we focus here on the integrity of the position, the relevant assets are only analyzed regarding the integrity property. For the relevant assets, risks are derived. The ISO 21434 standard proposes risk levels between one and five [17]. However, we use levels between zero and four in our approach as this simplifies further calculations.

Based on the identified risks, security controls are determined and integrated into the item definition of the TARA process. We distinguish here between controls whose existence is only known from design-time because they can not be validated during run-time (design-time controls) and controls that can be validated during run-time (run-time controls). For run-time controls, evidence is collected from the ECU, which is assessed based on these security controls. We assume that positive or negative evidence can be provided. Positive evidence shows that the control is active and in a valid state, i.e., it has been configured correctly, is working as expected, and has not been compromised. Negative evidence shows that a security control is not active or is not in a valid state.

The integration of security controls and the analysis of risk levels will be done in two steps. In the first step, only design-time controls are added to the item definition. These controls are either selected in the TARA process based on the identified risks or are already present in the ECU, as ECUs usually already have some controls implemented in their default setup. Based on all selected design-time controls, one TARA iteration is conducted, and a risk level is calculated for each risk. In the second step, run-time controls are analyzed. Run-time controls are also either selected based on the TARA process or are already included in the default setup of the ECU. In each TARA iteration, only the design-time controls and the currently analyzed run-time control are included in the item definition. Thus, each run-time control is analyzed in isolation to determine its effect.

Running Example: The output of the first step is shown in Table 8.1. The table was derived based on the risk assessment shown in Section 7.7. The table shows the identified risks (rows) and the corresponding security controls (columns). The first column represents the first step, in which the design-time controls are taken into account. The other columns represent the run-time controls. The values in the table are the corresponding risk levels. The last row shows the sum of all risk levels for the case that the respective control is implemented. Please note that no actual TARA was conducted to create the table but that the risks and corresponding risk levels are just examples for illustration purposes. This table only contains the risks and risk levels of the entity for which the trust is assessed. Furthermore, the table only contains the risks for the correspond-

ing scope (i.e. the data item to be forwarded from the ECU) and the property (e.g. integrity) to be protected. All other risks identified in the TARA process are not relevant for the trust assessment and are filtered out.

Table 8.1: Risks with the corresponding risk levels for the trust relationship between the vehicle computer and zonal controller

Risk	Original Risk	C1 - MACsec	C2 - Secure Boot	C3 - C.-F. Integrity	C4 - TLS	C5 - HSM
R1 (Compromised Application)	3	2	0	0	2	3
R2 (Compromised OS)	3	2	0	3	2	3
R3 (MitM external comm.)	4	1	4	4	1	4
R4 (Compromise Network Firmware)	2	1	2	2	1	2
R5 (MitM internal comm.)	1	1	1	1	0	1
R6 (Impersonation attack)	4	2	4	4	2	1
R7 (Physical attack on RAM)	1	1	1	1	1	1
Total Risk	18	10	12	15	9	15

Weight Calculation (Step 2)

To assess the trustworthiness of an ECU based on security controls, in the second step the importance of the run-time controls is determined. For this purpose, weights are assigned to each control. These weights are values between zero and one and are used to determine how much the trust opinion will be adjusted depending on the run-time evidence provided for this control. The weight for each security control is determined in three steps: First, the risk levels of all identified risks are summed up when only the design-time controls are applied. Second, the associated risk levels of all identified risks are summed up when the corresponding run-time control is applied. Third, a weight is calculated based on the difference between these two values. Depending on how many risks the corresponding run-time control affects and how much the risk levels are reduced, the more the control protects the system against identified risks. Thus, the more the risk levels in all risks are reduced, the higher is the weight for the control. Equation 8.2 shows the calculation of the weights. Here *C* represents the set of all security controls. The variable $sumRisks_{C_x}$ is the sum of all risk levels associated with the implementation of C_x, e.g. $sumRisks_{C_1}$ is the sum of risk levels if security control one is applied. $sumRisks_{\emptyset}$ is the sum of risk levels if no security controls are applied.

$$W_{C_x} = \frac{sumRisks_{\emptyset} - sumRisks_{C_X}}{\sum_{N=1}^{|C|}(sumRisks_{\emptyset} - sumRisks_{C_N})} \tag{8.2}$$

Running Example: The weight calculation for the security control MACsec is shown below. For the other security controls the approach is the same. Therefore, for the other controls only the weights are shown below.

$$W_{C_1} = \frac{18-10}{(18-10)+(18-12)+\cdots} = \frac{8}{8+6+3+9+3} = 0.28$$
$$W_{C_2} = 0.20\,,\ W_{C_3} = 0.11\,,\ W_{C_4} = 0.31\,,\ W_{C_5} = 0.10$$

Design-Time Trust Opinion Calculation (Step 3)

Based on the design-time information (DTI), i.e., the determined risk levels and the relevant run-time controls, ω_{DTI} is calculated. This opinion is used as a starting point for trust assessment based on run-time information and represents the minimal belief and minimal disbelief in the system based on DTI. This trust opinion is necessary because even if all design-time and run-time controls are active, there is still a remaining risk and therefore a non-zero disbelief in the system. On the other hand, the risk levels are already low in some systems because the design-time controls already reduce the risks. So even if no positive evidence for the run-time controls is provided during run-time, there could still be a belief in the system. These two aspects are reflected in ω_{DTI}.

DTI-based belief The DTI-based belief represents the minimum belief in the system. For this purpose, a worst-case analysis is conducted by calculating the maximum possible disbelief d_{max}. We assume full knowledge about all controls (none of the run-time controls are applied) so that the uncertainty is zero. To calculate d_{max}, the risk levels are analyzed when only the design-time controls are applied, as this is known from design-time, and all run-time controls provide negative evidence. Here, a combination of the maximum and the average risk levels of all risks is used. Based on the maximum risk level, the first d_{max} value is calculated. For example, when the maximum risk level is 4, this results in $d_{max} = 0.75$, as the d_{max} value is set to increase in 0.25 steps (see Equation 8.4). Then, a second d_{max} value is calculated, which increases linearly with the average risk level. For example, if the average risk level is 2, the second d_{max} value will be 0.5, as the average risk level is divided by the scaling factor $maxRisk = 4$. From these two values, the maximum value is selected.

DTI-based disbelief The DTI-based disbelief represents the minimum disbelief in the system. For this purpose, a best-case analysis is conducted. Here, the risk levels are analyzed for the case that the design-time controls are applied and positive evidence was provided for all run-time controls. The approach is the same as for the calculation of the maximum disbelief, with the difference that here the risk levels are taken into account for the case that all run-time controls are applied. This approach is represented in Equation 8.5.

Based on b_{DTI} and d_{DTI}, the DTI-based uncertainty u_{DTI} is calculated as shown in Equation 8.6.

$$b_{DTI} = 1 - d_{max} \tag{8.3}$$

$$d_{max} = max\{0.25 \times (maxRisk_{NoControls} - 1), avgRisk_{NoControls}/maxRisk\} \tag{8.4}$$

$$d_{DTI} = max\{0.25 \times (maxRisk_{AllControls} - 1), avgRisk_{AllControls}/maxRisk\} \tag{8.5}$$

$$u_{DTI} = 1 - b_{DTI} - d_{DTI} \tag{8.6}$$

Running Example: Based on the Equations 8.3, 8.4, 8.5, and 8.6, ω_{DTI} can be calculated. Using the risk values in Table 8.1 results in $d_{max} = 0.75$ and thus in $b_{DTI} = 0.25$. Furthermore, $d_{DTI} = 0.14$ and $u_{DTI} = 0.61$ can be calculated based on the risk values. Based on ω_{DTI} and run-time evidence, ω^{VC}_{ZC1} is calculated. If negative evidence is provided for all run-time controls, there is still a belief of 0.25, but a disbelief of 0.61 will be added to d_{DTI}. If positive evidence is provided for all controls, the disbelief is still 0.18, while 0.61 belief is added. This approach is described in more detail in the following.

Trust Opinion Calculation (Step 4 + 5)

The relevant run-time controls, the weights for each control, and ω_{DTI} are determined at design-time and stored in the Trust Model so that the TAF can access them at run-time.

Based on ω_{DTI} and depending on if positive or negative evidence is received for a control, either the belief or disbelief will be increased, and the uncertainty will be decreased by the same amount. How much the belief or disbelief will be increased depends on the specified weight of the control and u_{DTI} (see Equation 8.7). u_{DTI} is used in this equation as this is the uncertainty of the ECU, which is reduced with received evidence and thus knowledge about the ECU. For disbelief and uncertainty, the same equation is used as for the belief, as the value for these three attributes is the same. If no evidence is provided for a security control, e.g., because it could not be provided within the time requirements, neither belief nor disbelief will change. This results in a higher uncertainty for the final trust opinion as u_{DTI} has a high uncertainty.

$$\Delta b_{C_x} = \Delta d_{C_x} = \Delta u_{C_x} = W_{C_x} \times u_{DTI} \tag{8.7}$$

Running Example: We assume that positive evidence was provided for almost all run-time controls. Negative evidence was provided for control-flow integrity (C_3) as this control is not used in the corresponding application. Furthermore, no evidence was provided for HSM (C_5), so that we do not know if it is in a valid state. Based on the DTI-based trust opinion and the calculated weights, we can derive the trust opinion w^{VC}_{ZC1}. The amount of belief/disbelief added by each control is calculated with Equation 8.7 and is $\Delta b_{C_1} = \Delta u_{C_1} = 0.17$, $\Delta b_{C_2} = \Delta u_{C_2} = 0.12$, $\Delta d_{C_3} = \Delta u_{C_3} = 0.07$, and $\Delta b_{C_4} = \Delta u_{C_4} = 0.19$.

Based on these values, the trust opinion ω^{VC}_{ZC1} can be calculated:

- C1 - MACsec: Positive evidence
  $\omega_{DTI} = (0.25, 0.14, 0.61) \Rightarrow \omega_{C1} = (0.25 + 0.17, 0.14, 0.61 - 0.17) = (0.42, 0.14, 0.44)$
- C2 - Secure Boot: Positive evidence
  $\omega_{C1} = (0.42, 0.14, 0.44) \Rightarrow \omega_{C2} = (0.42 + 0.12, 0.14, 0.44 - 0.12) = (0.54, 0.14, 0.32)$
- C3 - Control-Flow integrity: Negative evidence
  $\omega_{C2} = (0.54, 0.14, 0.32) \Rightarrow \omega_{C3} = (0.54, 0.14 + 0.07, 0.32 - 0.07) = (0.54, 0.21, 0.25)$
- C4 - TLS: Positive evidence
  $\omega_{C3} = (0.54, 0.21, 0.25) \Rightarrow \omega_{C4} = (0.54 + 0.19, 0.21, 0.25 - 0.19) = (0.73, 0.21, 0.06)$

The trust opinion $\omega_{C4} = (0.73, 0.21, 0.06)$ is the final trust opinion ω^{VC}_{ZC1} which was calculated based on security controls as a trust source. The calculation of ω^{VC}_{ZC1} is also visualized in Figure 8.5.

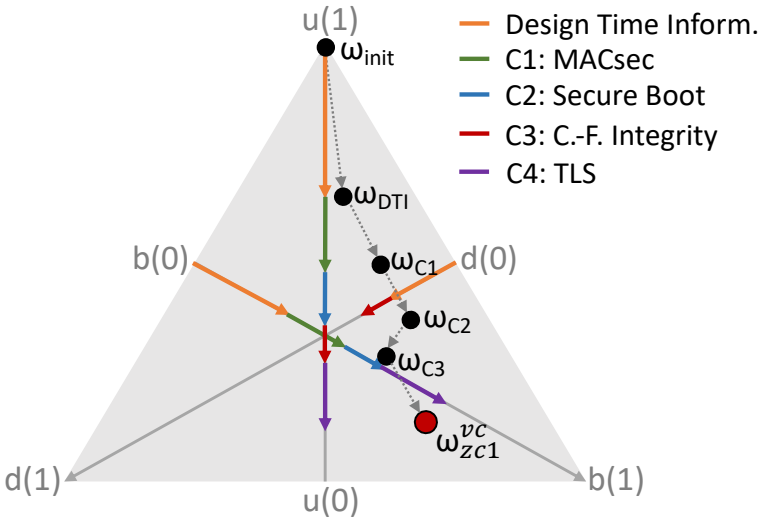


Figure 8.5: Calculation of a trust opinion based on implemented security controls.

8.3 Trust Assessment based on Trustworthiness Claims

As described in the previous section, the existence of security controls can be taken into account to determine the trustworthiness of a system. However, this approach is mainly focused on the in-vehicle network. To provide evidence about the integrity of a system to a remote entity, trustworthiness claims can be used. Trustworthiness claims are a data structure providing trustworthiness evidence about attributes pertinent to the ability of the vehicle or MEC to transmit correct data in the V2X messages. These are signed items describing the state of a given component. The state can be, e.g., the integrity of the component or its configuration. The Trustworthiness Claims are sent as part of the CAMs, CPM and misbehavior reports. When the claims are received by another node, they are processed by the Attestation and Integrity Verification (AIV) component and are then forwarded to the TSM. The trustworthiness evidence can be placed into several categories, which are shown below. For more detailed information regarding trustworthiness claims, we refer to Deliverable 5.1 [12].

- Design-time integrity: Cryptographic capabilities each entity can support.
- Boot-up integrity: Information whether the device has a secure boot mechanism.
- Run-time integrity: Information whether the device has capabilities to perform introspection on running processes and if these checks failed.
- Communication integrity: Information whether the cryptographic keys are correctly installed and used.
- Certified application attribute: Information whether the processing was done by certified services and functions.

The entity receiving the trustworthiness claims can use this information as a trust source and calculate a trust opinion on the device that provided these trustworthiness claims. As with the other trust sources, a trust source quantifier template is necessary for this trust source. This trust source quantifier template has to take into account the single categories and the provided

evidence to create a trust opinion on the external entity which provided the evidence. Alternatively, a trust source quantifier template is created for each of the categories. In this case, a trust opinion is created for each category, which are then fused together to a final trust opinion. How exactly this trust source quantifier template quantifies the received input, is ongoing research and will be elaborated in the context of the CONNECT project.

Chapter 9

Required Trustworthiness Level (RTL)

This chapter introduces Required Trustworthiness Level (RTL), including its meaning and its impact, as well as a method to calculate its believe part. Uncertainty and disbelief are shortly discussed as well, but a method to calculate them is not in the scope of this deliverable, nonetheless it will be part of the deliverable D3.3.

9.1 RTL definition

Required Trustworthiness Level (RTL) represents the required level of trustworthiness in an entity and is calculated at design-time. From a technical perspective, RTL defines the minimal level of trustworthiness to consider a given vehicle function or data created or received by a vehicle reliable for the system. Its definition may be based on cybersecurity risk analysis factors, technical demands or demands forced by requirements to obtain technical approval to produce the car, for example. These technical requirements can be translated to trust requirements, and from this deliverable we are going to show alternatives to set RTL based on what is already know during design, starting with a risk-based subjective opinion representation of the RTL.

Based on subjective logic, RTL is composed by belief, disbelief and uncertainty values. RTL sets the minimum required belief, and the maximum acceptable uncertainty and/or disbelief. The Figure 9.1 shows the boundaries created when defining this triad and where a trust opinion is expected to be located within the intersection between them.

RTL is defined at design time and it is used as a trust decision threshold. From a design perspective, RTL can not rely on actual evidence, since the vehicle is under development yet, nonetheless it is possible to take advantage of risk assessment tools, assumptions and considerations taken, and observing expected cybersecurity scenarios.

A risk-based approach was developed to determine RTL, taking advantages of the TARA, which is already standardised by [19] and executed by vehicle manufactures as part of their development process in order to assess the cybersecurity risks. The following section introduces this approach applied to belief calculation. The definition of uncertainty and disbelief is not part of this document, but a method will be provided in the deliverable D3.3. This is one approach of how RTL can be represented and calculated. The next deliverable will also explore different alternatives to express RTL, which may include probabilistic representations or distribution functions, for example. These new possible representations will be considered for evaluating the method, and enabling the creation of more effective methods, taking into account pros and cons, or different

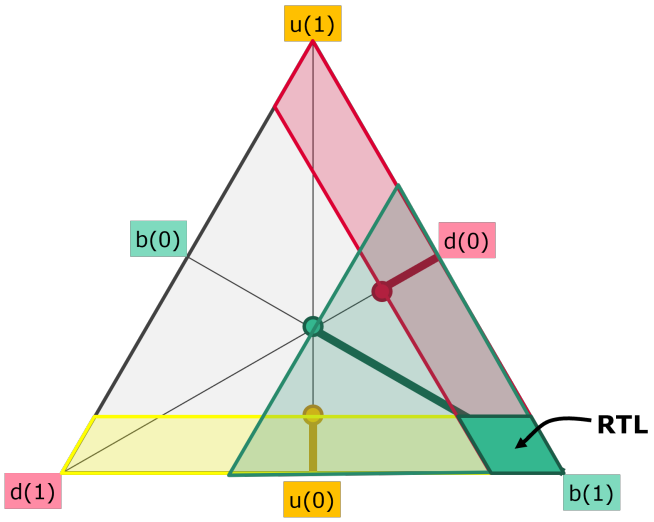


Figure 9.1: Graphical representation of RTL within subjective logic triangle

cases where more specificity is required or there is little information that can be used to calculate RTL as presented below. The comparison with ATL is also a concern and need to be aligned which representation of RTL and ATL is comparable.

9.2 RTL calculation

The purpose of this section is to introduce a method for estimating a minimum necessary belief based on risk analysis. Nonetheless, we also provide certain factors that might impact or be considered for further uncertainty and disbelief calculation.

First, it is important to revisit some subjective logic definitions and put them into the RTL context, to explain what they mean for the RTL context. Uncertainty refers to a lack of information in which there is insufficient trust evidence to rely on or to confirm whether a trustee is trustworthy. During design time, factors such as technical model system completeness, assumptions, level of rigor for testings may be used to estimate uncertainty. Changes on the system model can also increase or decrease uncertainty, by introducing new unknown or not considered components from task offloading, for example.

Disbelief refers to the features of a system or the connection between the trustee and trustor that infers the system is not trustworthy, often known as negative evidence. Using a risk-based approach, disbelief might be estimated based on residual threats. In other words, if certain threats are predicted and a decision is taken to maintain the risk, or if any residual risk remains after applying mitigation strategies, this may be classified as negative evidence.

In terms of belief, positive evidence is employed to build belief in the trustee. Secured connections and protocols, authentication methods, physical protection, and a variety of other features can all be used to create a trustworthy system. According to our risk-based approach, the riskier the system is, the greater the needed belief value. In the next section, the risk-based approach to calculate the belief part of the RTL is presented. This technique considers TARA output such as attack feasibility and impact rating to calculate.

9.2.1 Risk-based belief calculation

Belief is built from positive evidence, however, there is no pieces of evidence available during design. In this case, we explore known risks from TARA to determine how much belief is needed to make the trustee trustworthy. The introduced method map the risk level into belief values.

To have a first understanding, let's consider two comparable systems, which can provide the same kind and amount of trust sources. The system A has many risks classified as 5 due to lack of security features. The system B has only risks classified as 2 because there is cybersecurity features against the majority of the expected threats. In this example, the system A need to have more pieces of evidence to attest its trustworthiness since its risk level is 5. On the other hand, the second system does not need to have as the same amount of evidence as A, considering its risk level. The risk-based approach considers the item risk level to determine how much trust is needed in order to build trust.

To calculate belief, we take advantages of the already introduced automotive Threat Analysis and Risk Assessment (TARA) in the section 7.2.1. TARA is a standardised framework implemented throughout vehicle design time [19]. This allows cybersecurity engineers to anticipate cybersecurity threats and give support to create strategies to minimize or mitigate their risk over the system. TARA has as output a list of threats and damage scenarios for a given technical system model, including *attack feasibility* (F), *impact rating* (I), and *risk level* (R) calculated from I and F.

The Figure 9.2 summarises the steps to calculate belief from the list of risks and associated attack feasibility and impact ratings.

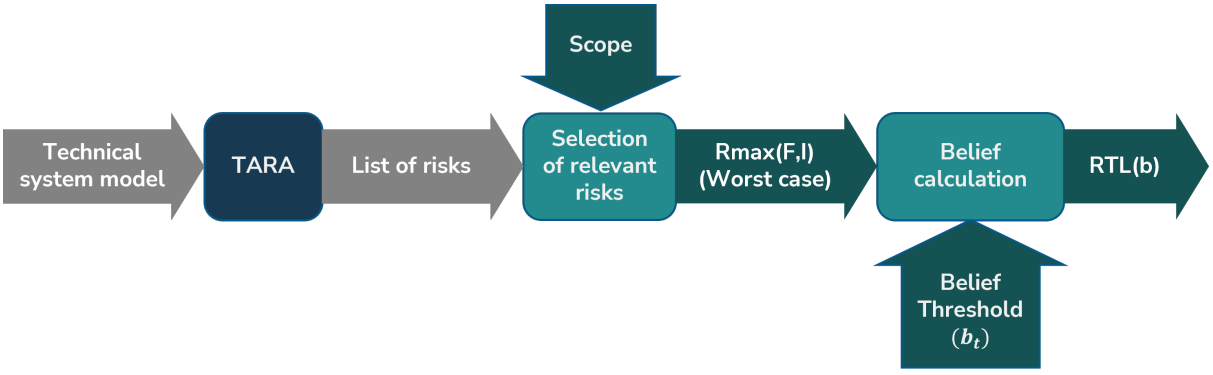


Figure 9.2: Risk-based belief scheme calculation

The first step is to use the output of TARA to access the list of expected risks. TARA can highlight a vast list of risks and potential threats, but they are related to the overall item and some of them might be irrelevant to the considered scope. Following that, it is required to elect the most relevant scope-related risks to be basis for the further belief calculation. The suggested technique considers the worst-case scenario to ensure that the RTL will cover all scenarios, nonetheless another criteria can be created based on a specific company needs or deficiency of providing further cybersecurity evidence once the vehicle is running. In this document, we follow the worst-case scenario criteria.

From the scope-relevant risk list, we select the worst case considering the highest risk level. The highest risk level is considered for the belief calculation, in addition to the *Belief Threshold* b_t, which seeks to create a minimal required belief in circumstances when the worst-case scenario has attack feasibility very low or the impact rating negligible. b_t also impacts higher risk levels, but its impact is getting softer when handling with higher risk levels, as shown in Figure 9.3.

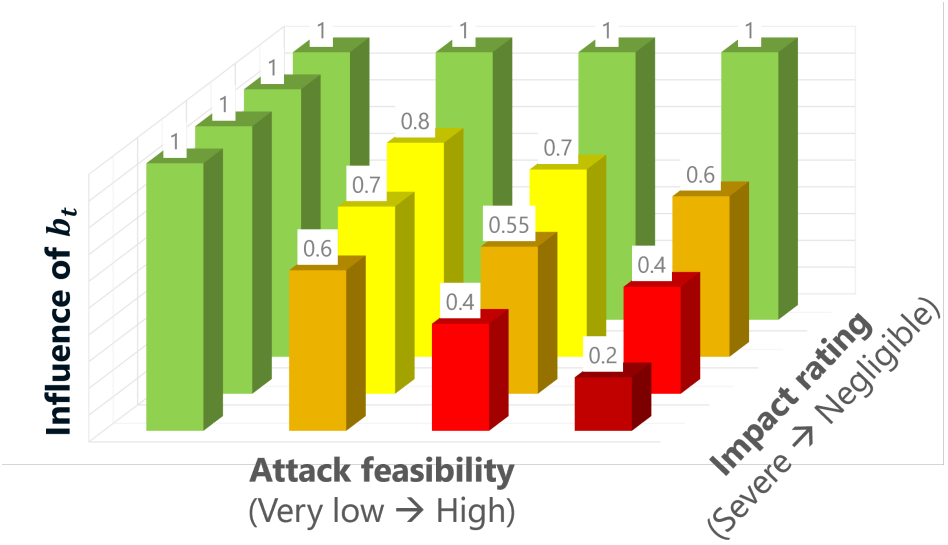


Figure 9.3: Influence of belief threshold

The figure 9.3 can be interpreted as a percentage level of interference of b_t according to risk level. For negligible impact rating and/or very low attack feasibility it has 100% of interference on the belief value, on the other hand, it has 20% of influence on risks with severe impact and high feasibility. A method for defining b_t is not clear yet, but it might take into consideration safety and ethical aspects, internal company decisions, or to increase rigor based on non-security demand, for instance. For this deliverable, it is considered as an assumption by the vehicle manufacturer based on its cybersecurity strategy.

In order to map the belief from the risk level, the following mathematical equations are used:

$$\Delta = \frac{1 - b_t}{5}, 0 \leq b_t \leq 1 \tag{9.1}$$

Where Δ is the difference between levels, b_t is the set belief threshold, a number between 0 and 1, and 5 is regarding to the number of risk levels set by TARA.

$$b_{RTL} = b_t + ((R_{max}(F, I) - 1) \times \Delta) \tag{9.2}$$

Where b_{RTL} represents the belief part of the RTL, and $R_{max}(F, I)$ is the risk level for the worst case, ranging from 1 to 5,. The risk function can vary between manufacturers, according to [19].

Equation 9.2 converts the risk level, which has 5 levels, into a belief value, which has a range of 0 to 1. The equation has two significant variables: $R_{max}(F, I)$ and b_t. The car manufacturer determines the risk level calculation ($R_{max}(F, I)$), which may be tailored to each company's specific needs. [19] standardized the risk level from 1 to 5, but for the equation 9.2, we utilize it from 0 to 4, for simplicity, which is why we reduce 1 from $R_{max}(F, I)$. b_t is chosen by the company and context-related, including subjective factors and isolated aspects. For example, to set a minimum belief for edge scenarios, where without a threshold, might accept any belief value, including 0. This deliverable will not go into details about determining b_t, this will be the focus of the next deliverable. Now we consider b_t as an assumption made by the manufacturer and this can take into consideration safety aspects, regulations or any other relevant characteristics that is not covered by TARA.

9.2.2 Running example - Risk-based belief calculation

The architecture shown in the Figure 9.4 is used to demonstrate the proposed method and it is a simplified version of the Figure 3.1. The use case is the transmission of a vehicle location data from the Global Navigation Satellite System (GNSS) and transmitted across the network to the vehicle computer, passing through the GNSS Electronic Control Unit (ECU), responsible for receiving the GNSS data, and the Zonal controller 1. The location data goes through an unprotected communication channel. The use case scope is focused on the integrity of the transmitted location data from the GNSS to the vehicle computer.

The trust network is created between the vehicle computer(trustor) and the GNSS data (trustee).

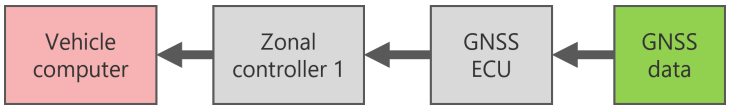


Figure 9.4: Simplified running example architecture

From the TARA presented in the Chapter 7.7, we can analyse the table 7.15, where is listed the risk levels for the running example. To calculate the RTL belief, we need to consider the scope to select the relevant list risk. Based on the required integrity of the data, the related damage scenarios are DS.2, DS.3 and DS.4, from the Table 7.16. Consequently, the damage-scenario associated threats are TS.4, TS.5 and TS.6, from the table 7.11. The selected relevant list of risks is summarised in the Table 9.1, including the risk levels based on safety (S), financial (F), operational (O) and privacy (P) impacts.

ID	Threat scenario	Damage scenario	S	F	O	P
R.5	TS.4	DS.2	3	1	*4*	3
R.6	TS.5	DS.2	1	1	1	1
R.7	TS.5	DS.3	1	1	1	1
R.8	TS.6	DS.2	2	1	*2.5*	2
R.9	TS.6	DS.4	2	2	*3*	1

Table 9.1: List of relevant risks

Taking a look at the Table 9.1, it is noticeable that the highest risk level is R.5, with operational risk equal 4 and safety risk equal 3. For this example we consider the highest risk (operational) as risk level, following the same approach of the example provided in [19]. With that, the highest relevant risk for the scope $R_{max}(F, I)$ is 4, referred to R.5.

For this use-case, we are going to assume a belief threshold (b_t) equal 0.5, in other words, the consider that the vehicle manufacture has decided that the minimum acceptable belief must be equal or higher than 0.5 for any case.

To calculate b_{RTL}, we need to calculate Δ first, by equation 9.1:

$$\Delta = \frac{1 - b_t}{5} = \frac{1 - 0.5}{5} = \mathbf{0.1} \tag{9.3}$$

Replacing Δ in the equation 9.2, we have:

$$b_{RTL} = b_t + ((R_{max}(F, I) - 1) \times \Delta) = 0.5 + (4 - 1) \times 0.1) = \mathbf{0.8} \tag{9.4}$$

For the given example, the minimum acceptable belief is 0.8. In other words, the TAF needs to collect pieces of evidence to reach at least belief equal 0.8 to consider the GNSS data trustworthy. The 0.2 left is composed by uncertainty and disbelief values. Mitigation strategies can make the required belief lower, by solving cybersecurity issues and implementing cybersecurity features to reduce those found risk.

Chapter 10

Conclusion

This document describes the current state of the design of the Trust Assessment Framework and all its related components as well as its interaction with the risk assessment framework.

Following the three phase approach laid out in Deliverable 3.1, we have specified the architecture, the components and interfaces and the functionality of the standalone TAF corresponding also to the prototype implementation of this TAF version.

Furthermore, we also describe functionality and three different cases how a federated TAF is to be implemented. A final and detailed architecture is due only for Deliverable D3.3 along with an implementation of the federation functionality as needed by the use-cases.

Finally, this deliverable discusses requirements and challenges for a TAF version acting as a digital twin, the TAF-DT. Again, a final and detailed architecture is due only for Deliverable D3.3 along with an implementation of the federation functionality as needed by the use-cases. For this functionality, the TAF-DT will rely on mechanisms designed and implemented in Work Package 5.

With this achievement, Work Package 3 concludes its contributions to Milestones 1-5.

Chapter 11

List of Abbreviations

AC	Attack Complexity
ACC	Adaptive Cruise Control
AIV	Attestation and Integrity Verification
ATL	Actual Trustworthiness Level
AV	Attack Vector
CFI	Control Flow Integrity
CIV	Configuration Integrity Verification
CRL	Cumulative Risk Level
CVSS	Common Vulnerability Scoring System
DAG	Direct Acyclic Graph
DLT	Distributed Ledger Technology
ECU	Electronic Control Unit
FIR	Federation Instance Request
FIRST	Forum of Incident Response and Security Teams
FR	Federation Request
GNSS	Global Navigation Satellite System
HARA	Hazard Analysis and Risk Assessment
IDS	Intrusion Detection Systems
LDM	Local Dynamic Map
MEC	Mobile Edge Computer
NDQ	Node Discovery Query
NVD	National Vulnerability Database
OIP	Opinion Information Point
PR	Privileges Required
RA	Risk Assessment
RC	Report Confidence
RFRA	Request For Risk Assessment
RL	Remediation Level
RTL	Required Trustworthiness Level
SIEM	Security Information and Event Management
SR	Security Requirements
STN	Subjective Trust Network
TAF	Trust Assessment Framework

TAM	Trust Assessment Manager
TAQ	Trust Assessment Query
TAR	Trustworthiness Assessment Request
TARA	Threat Analysis and Risk Assessment
TAS	Trust Assessment Service
TC	Trustworthiness Claim
TD	Trust Decision
TDE	Trust Decision Engine
TMI	Trust Model Instance
TMT	Trust Model Template
TSM	Trust Sources Manager
TSR	Trust Source Request
UI	User Interaction
VC	Vehicle computer

Appendix A

CVSS attributes

Value			Description	Score
Base Metric	Attack Vector	Physical (P)	The susceptible system must be physically accessible to the attacker (e.g., firewire attacks).	0.20
		Local (L)	The attacker needs a local account on the target system (e.g., privilege escalation attack).	0.55
		Adjacent (A)	The attacker must have access to the same broadcast or collision domain as the vulnerable system (e.g., ARP spoofing, Bluetooth attacks).	0.62
		Network (N)	The vulnerable interface operates at layer 3 or higher in the OSI network stack, making it remotely exploitable (e.g., remote buffer overflow in a network service).	0.85
	Attack Complexity	High (H)	There are specialized conditions, such a race condition with a small window or a need for social engineering techniques that would be obvious to experienced people.	0.44
		Low (L)	No unique circumstances, such as when the system is accessible to a large number of users or the susceptible configuration is common, exist for exploiting the vulnerability.	0.77
	Privileges Required	High (H)	The exploit involves specialized conditions or social engineering techniques that would be evident to experienced individuals.	0.27
		Low (L)	The vulnerability can be exploited under common circumstances without the need for specialized conditions or user interaction.	0.62
		None (N)	The attacker can initiate the attack without first authenticating.	0.85
	User Interaction	None (N)	The attacker does not need to authenticate in order to initiate the attack.	0.85
		Required (L)	The user must take action in order to successfully exploit this vulnerability before it can be used against them.	0.62

Temporal Metric	Exploit Code Maturity	Not Defined (X)	This value is assigned when there is not enough information to select one of the other values. It has the same scoring impact as assigning the value High and has no effect on the total Temporal Score.	0.0
		Unproven (U)	Either an exploit is hypothetical or there is no exploit code accessible.	0.91
		Proof-Of-Concept (P)	For the majority of systems, an attack demonstration is not feasible even when proof-of-concept exploit code is available. The code or technique may need to be significantly modified by a knowledgeable attacker because it is not always effective.	0.94
		Functional (F)	There is working exploit code available. In most cases when the vulnerability exists, the code functions.	0.97
		High (H)	There is functional autonomous code, no need for an exploit (manual trigger), and information is readily available. Exploit code operates under all circumstances or is actively disseminated by an autonomous agent (such as a worm or virus). Systems linked to the network are susceptible to scanning or exploitation efforts. Exploit development has advanced to the point that automated tools are trustworthy, accessible, and simple to use.	1.0
	Remediation Level	Not Defined (X)	This value is assigned when there is insufficient information to select one of the other values. It has the same scoring impact as assigning Unavailable and has no effect on the overall Temporal Score.	0.0
		Official Fix (O)	There is a full vendor solution available. Either an official fix has been released by the vendor, or an upgrade is available.	0.95
		Temporary Fix (T)	There is a legitimate, short-term treatment available. Instances where the vendor releases a temporary patch, tool, or workaround are included in this.	0.96
		Workaround (W)	Unofficial, non-vendor solutions are accessible. Some users of the impacted technology will develop their own patches, offer solutions to get around the issue, or take other actions to minimize it.	0.97
		Unavailable (U)	Either there isn't a solution, or applying it is impossible.	1.0
	Report Confidence	Not Defined (X)	This value is assigned when there is not enough information to select one of the other values. It has the same scoring impact as Confirmed and has no effect on the total Temporal Score.	0.0
		Unknown (U)	Impacts have been reported, which suggests a vulnerability exists. The reports claim that the vulnerability's cause is unknown, or that the cause or effects of the vulnerability may vary between studies.	0.92

		Reasonable (R)	Substantial information is made public, but researchers lack complete confidence in the underlying cause or lack access to the source code necessary to thoroughly verify all of the interactions that may have contributed to the outcome. Yet, there is a reasonable level of confidence that the defect can be reproduced and that at least one impact can be validated (proof-of-concept exploits may provide this).	0.96
		Confirmed (C)	There are detailed reports available, or a working reproduction is feasible (functional exploits may provide this). The researcher's claims can be independently verified with access to the source code, or the vulnerability has been confirmed by the creator or vendor of the affected code.	1.0
Environmental Metric	C.I.A. Triad	Not Defined (X)	This value is assigned when there is not enough information to select one of the other values. Assigning this value has the same scoring impact as applying the value Medium and has no effect on the overall Environmental Score.	0.0
		Low (L)	The organisation or those connected to the organization will certainly suffer catastrophic consequences if [Confidentiality — Integrity — Availability] are lost (e.g., employees, customers).	0.5
		Medium (M)	The organisation or those connected to the organization will certainly suffer catastrophic consequences if [Confidentiality — Integrity — Availability] are lost (e.g., employees, customers).	0.5
		High (H)	The organisation or those connected to the organisation are likely to experience just a little negative impact from the loss of [Confidentiality — Integrity — Availability] (e.g., employees, customers).	1.5

Table A.1: CVSS v3.1 characteristics

Bibliography

[1] Common Attack Pattern Enumeration and Classification (CAPEC). https://capec.mitre.org/. Accessed: February 14, 2024.

[2] Common Platform Enumeration (CPE). https://nvd.nist.gov/products/cpe. Accessed: February 14, 2024.

[3] Common Vulnerabilities and Exposures (CVE). https://cve.mitre.org/. Accessed: February 14, 2024.

[4] Common Weakness Enumeration (CWE). https://cwe.mitre.org/. Accessed: February 14, 2024.

[5] Drools Specification 8.44.0. https://docs.drools.org/8.44.0.Final/drools-docs/drools/introduction/index.html. Accessed: February 14, 2024.

[6] FIRST - Forum of Incident Response and Security Teams. https://www.first.org/. Accessed: February 14, 2024.

[7] STRIDE Threat Model. https://learn.microsoft.com/en-us/azure/security/develop/threat-modeling-tool-threats#stride-model. Accessed: February 14, 2024.

[8] ECE/TRANS/WP.29/2020/79 Revised: UN Regulation on uniform provisions concerning the approval of vehicles with regard to cyber security and of their cybersecurity management systems. http://www.unece.org/DAM/trans/doc/2020/wp29grva/ECE-TRANS-WP29-2020-079-Revised.pdf, 2020. Accessed: February 14, 2024.

[9] ISO/SAE 21434:2021. Road vehicles Cybersecurity engineering. *ISO/TC 22/SC 32 Technical Standard*, 2021. https://www.iso.org/standard/70918.html.

[10] The CONNECT Consortium. Architectural specification of connect trust assessment framework, operation and interaction. Deliverable D3.1, October 2023.

[11] The CONNECT Consortium. Conceptual architecture & customizable tee and attestation models specification. Deliverable D4.1, October 2023.

[12] The CONNECT Consortium. Distributed processing and ccam trust functions offloading & data space modelling. Deliverable D5.1, August 2023.

[13] The CONNECT Consortium. Operational landscape, requirements and reference architecture - initial version. Deliverable D2.1, October 2023.

[14] The CONNECT Consortium. Connect trust & risk assessment & cad twinning framework (final version). Deliverable D3.3, February 2025.

[15] The CONNECT Consortium. Integrated framework (final release), use case evaluation and project impact assessment. Deliverable D6.2, August 2025.

[16] ISO. Iso 26262-3:2018 - road vehicles - functional safety - part 3: Concept phase. page 28. pub-ISO, 2018.

[17] ISO. Iso/sae 21434:2021. page 88. pub-ISO, 2021.

[18] ISO/IEC. Iso/iec 30141:2018 - internet of things (iot) - reference architecture. page 81. pub-ISO, 2018.

[19] ISO/SAE. 21434 - road vehicles — cybersecurity engineering, 08 2021.

[20] Audun Jøsang. *Subjective logic*. Springer, 2016.

[21] Audun Jøsang, Stephen Marsh, and Simon Pope. Exploring different types of trust propagation. In *International Conference on Trust Management*, pages 179–192. Springer, 2006.

[22] Joseph Kamel, Mohammad Raashid Ansari, Jonathan Petit, Arnaud Kaiser, Ines Ben Jemaa, and Pascal Urien. Simulation framework for misbehavior detection in vehicular networks. *IEEE Transactions on Vehicular Technology*, 69(6):6631–6643, 2020.

[23] National Institute of Standards and Technology (NIST). National Vulnerability Database (NVD). https://nvd.nist.gov/. Accessed: February 14, 2024.

[24] National Institute of Standards and Technology (NIST). NIST Special Publication 800-39: Managing Information Security Risk: Organization, Mission, and Information System View. Special Publication 800-39, NIST, 2011. Accessed: February 14, 2024.

[25] National Institute of Standards and Technology (NIST). NIST Special Publication 800-30 Rev. 1: Guide for Conducting Risk Assessments. Special Publication 800-30, NIST, 2012. Accessed: February 14, 2024.

[26] The Forum of Incident Response and Security Teams (FIRST). Common Vulnerability Scoring System (CVSS) Version 3.1. https://www.first.org/cvss/v3.1/specification-document, 2019. Accessed: February 14, 2024.