PRISSMA Project
Plateforme de Recherche et d'Investissement pour la Sûreté
et la Sécurité de la Mobilité Autonome
04/2021 - 04/2024

[L5.5] RELIABILITY OF AUTONOMOUS VEHICLE SYSTEMS IMPLEMENTATIONS

[L5.5] RAPPORT SUR LA FIABILITE DE L'ECOSYSTEME VEHICULAIRE AUTONOME SECURISE.

Main authors: B. Bannour (CEA) and S. Haddad (OPPIDA)

Abstract. This deliverable is associated with Task 5.5, which addresses reliability concerns in connected autonomous vehicular ecosystems through protocol verification methodologies. CEA presents a tooled Verification and Validation (V&V) methodology based on Interaction Models (IM), resembling UML sequence diagrams or Message Sequence charts. These models capture communication flows in distributed systems and protocols. The methodology integrates IMs with Runtime Verification (RV) techniques to evaluate protocol implementations' adherence to specifications during testing or operation. We showcase the methodology's application through the analysis of V2V/V2I/V2X secured communication using ETSI TS 103 900 Cooperative Awareness Messages (CAM).

Résumé. Ce livrable est associé à la tâche 5.5, qui aborde les problèmes de fiabilité dans les écosystèmes véhiculaires autonomes connectés grâce à des méthodologies de vérification de protocoles. CEA présente une méthodologie outillée de vérification et de validation (V&V) basée sur des modèles d'interaction (IM), ressemblant à des diagrammes de séquence UML ou à des MSC (Message Sequence charts). Ces modèles capturent les flux de communication dans les systèmes distribués et les protocoles. La méthodologie intègre les IM avec des techniques de vérification à l'exécution (RV) pour évaluer la conformité des implémentations de protocoles aux spécifications lors des tests ou de l'exploitation. Nous illustrons l'application de la méthodologie à travers l'analyse de la communication sécurisée V2V/V2I/V2X basé sur des messages CAM (Cooperative Awareness Messages) tels que standardisés par l'ETSI TS 103 900.

1 INTRODUCTION

This deliverable provides an overview of the activities conducted in Task 5.5. The primary goal of this task is to explore the reliability issues of autonomous vehicular ecosystems through the description of methodology(ies) for verifying protocols (verification of the properties of these protocols) associated with connectivity in autonomous vehicular ecosystems.

CEA has developed a tooled Verification and Validation (V&V) methodology based on Interaction Models (IM) akin to UML sequence diagrams or Message Sequence Charts. These models are specifically tailored to represent communication flows, particularly within distributed systems and communication protocols. The methodology combines interaction models and Runtime Verification (RV) techniques that can be used to analyze communication protocols implementations adherence to their specification at testing stages or at the operation phase.

We showcase the methodology applied to the analysis of V2V/V2I/V2X secured communication based on ETSI's Cooperative Awareness Messages (CAM) defined in ETSI TS 103 900 which aim to achieve information sharing between nearby vehicles and infrastructure to improve road safety while ensuring message authenticity and integrity.

As identified in PRISSMA deliverable 5.2 and 5.3, AI based autonomous systems are large and complex systems that rely on the correct and secured exchange of information between its different components to provide a global safe and efficient road transport service to its users. For the different AI based components of the system to perform correctly, they need to trust the different communication means with the different sources from which they gather information to perform correctly. For that not only security requirements have to be made on those communications (cf. PRISSMA deliverable 5.3 section 3.2), but also implementation and designed have to be evaluated. The tool presented in section 3 and 4 as well as the case study evaluations presented in section 4, demonstrate how higher level of assurance can be reached thanks to these advanced V&V technics.

2 BEHAVIORAL SPECIFICATION USING INTERACTION MODELS

Interactions such as UML Sequence Diagrams and Message Sequence Charts are broadly adopted models for describing behaviors of communication protocols and distributed systems. They focus on specifying message exchanges between subsystems with rich choice and scheduling operators, and they are associated with an intuitive graphical depiction.

The behavior of the system can be further formalized using various behavioral languages (automata, behavior trees, process algebras etc). In the following, we use interaction models to describe the intended behaviors of the communication protocols and distributed systems.

Interactions are models whose most well-known instances are UML Sequence Diagrams (UML-SD) [1] or Message Sequence Charts (MSC) [2]. Interactions specify the communication flow between entities constituting a system. They are particularly adapted to specify distributed system behaviors, as distributed systems are, by nature, composed of subsystems interacting via message transmissions. They are associated with graphical representations. This kind of representation fits in with several intuitions commonly shared by software engineers: time flows from top to bottom along lifelines, message passing is represented by horizontal arrows, scheduling operators are visualized by annotated boxes.

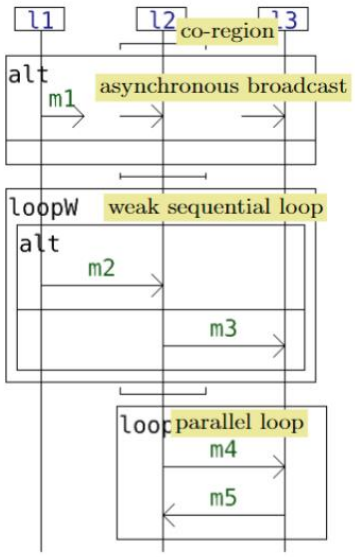


Figure 1 Interaction example

In Figure 1, we present the graphical representation of an interaction, generically denoted by *i*. Functionalities offered by interaction-based V&V tool are based on our works [3] [4] in which we define semantics of interactions without the need for translations to other formalisms. The interaction-based V&V tool computes part of this semantics, based on the key notion of *follow-up interaction*.

Figure 2 illustrates the four possible follow-up interactions that can be reached by performing a single execution step of the interaction *i*. An execution step consists in executing one action at a time, and is written as $i \xrightarrow{a} i'$ where *i'* is the follow-up interaction upon the action *a*.

In a nutshell, *i'* specifies possible continuations, according to *i*, after the occurrence of the action *a*. Such steps can then be displayed as graphical execution tree by the interaction-based V&V tool as shown in Figure 2.

Being grounded by this small-step semantics, the interaction-based V&V tool is, to the best of our knowledge, the first tool to offer interaction animation without going through translation mechanisms to intermediate formalisms like automata, petri-nets or others.

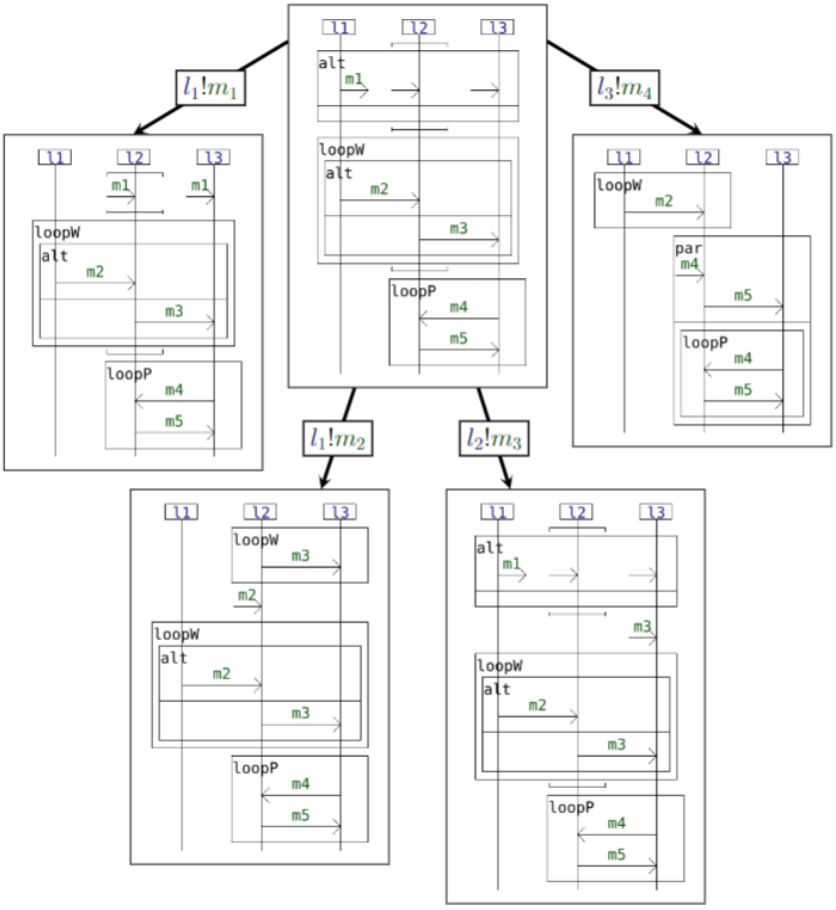


Figure 2 Follow-ups of the interaction from Figure 1

2.1 Interaction language

The first basic elements of an interaction model are the lifelines and the messages exchanged between them. Let us consider a set *L* of lifelines: a subsystem of the Distributed System (DS) is then described by a subset of the lifelines, possibly reduced to one. Let us then consider a set *M* of all the messages that can transit through the DS.

Atomic actions are of the form *l!m* (resp. *l?m*) corresponding to the emission (resp. reception) of the message *m* in *M* from (resp. by) the lifeline *l* in *L*.

Traces are sequences of actions where ε represents the empty sequence and are concatenated using the "." operator. Semantics of interactions are given in terms of traces.

As described in [3], interaction models correspond to expressions built over the empty interaction, denoted by Ø, with the empty trace as the only accepted trace and actions (of the form

l!m or *l?m*) with a trace reduced to a single action as the only accepted trace. We then use operators to compose interactions into more complex expressions. The *strict*, *seq*, *par* and newly introduced *coreg* constructs are binary operators which schedule the execution of actions w.r.t. one another. Let us consider two interactions i_1 and i_2. *strict* stands for strict sequencing and a behavior of *strict*(i_1,i_2) is such that a behavior from i_1 must be entirely expressed before any action from i_2 can occur. *par* stands for parallel or interleaved behaviors so that actions occurring in i_1 and i_2 can occur in any order in behaviors expressed by *par*(i_1,i_2). *seq* stands for weak sequencing and is in essence the key operator for interactions. With *seq*, interleaving is only possible between actions that occur on different lifelines, i.e. the behaviors of *seq*(i_1,i_2) are defined as with the *strict* operator for actions occurring on the same lifeline (whatever it may be) and as with the *par* operator for actions occurring on different lifelines.

A behavior of *alt*(i_1,i_2) is either a behaviour of i_1 or i_2 according to a non-deterministic and exclusive choice between the two alternatives.

The *coreg* construct behaves as *par* on certain specific lifelines and as *seq* on the others. It is used as *coreg(L')*(i_1,i_2)$ with *L'* a subset of *L*, the set of lifelines on which the co-region behaves as *par*.

The unary operators *loopS*, *loopW*, and *loopP* characterize repetitions (0, 1 or more times) of behaviors according to a certain scheduling policy which is resp. *strict*, *seq* and *par*. Interactions are both described as a textual expression in the input syntax of the interaction-based V&V tool and visualised by drawings that wide use of UML-SD and MSC has made familiar to many software engineers (see Figure 1).

2.2 Execution tree

Each interaction *i* characterises a set $\sigma(i)$ of accepted traces, i.e. finite sequences of actions defined on lifelines occurrring in *i*. This set of accepted traces can be defined in an operational-style using either inductive rules (in the style of Plotkin) or through the definition of an execution function, denoted by χ as in [4].

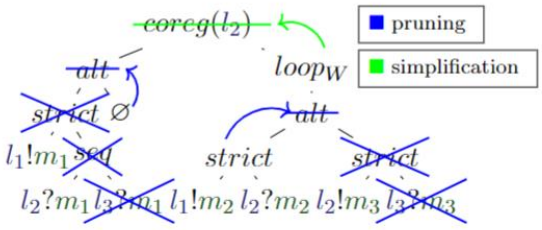


Figure 3 Pruning when executing $l_3!m_4$

From an initial interaction *i*, we may execute an action *a* which is immediately executable. Which actions this concerns can be determined statically via a frontier function *front*. *front(i)* gives the set of immediately executable actions as the set of positions where these actions appear in the expression denoting *i*. Via their recursive definition, interactions have a tree-like structure, as illustrated on Figure 3. Those trees are binary-trees and we can pinpoint unambiguously each sub-tree via its position as a word *p* in $\{1,2\}^*$. More precisely, 1 (resp. 2) allows to access to the left direct subinteraction or the unique direct interaction (resp. the right direct subinteraction). For any interaction *i*, *pos(i)* designates the set of its positions, and, for any *p* in *pos(i)*, $i_{|p}$ designates the sub-interaction at position p. For example, for the interaction *i* =

seq(alt($l_1!m_1$,$l_2?m_2$),$l_1!m_3$), $i_{|1}$ is the interaction *alt*($l_1!m_1$,$l_2?m_2$), $i_{|12}$ is the interaction $l_2?m_2$ and $\{\varepsilon, 1,2,11,12\}$ is the set of positions of *i*. Thus, *front(i)* is a subset of *pos(i)* contains the positions of all actions that can start a trace in $\sigma(i)$. The execution of an action $a = i_{|p}$ with *p* in *front(i)* yields a follow-up interaction *i'* (obtained with $\chi(i,p)$ accordingly to notations of [4]) which set of accepted traces, i.e. $\sigma(\chi(i,p))$, exactly contains the traces of that of the original interaction *i* that start with the specific action at position *p* in the interaction *i*.
Finally, we have

$$\sigma(i) = \begin{cases} \{\epsilon\} \cup \bigcup_{p \in front(i)} i_{|p}.\sigma\big(\chi(i,p)\big) & \text{if } \epsilon \text{ is accepted by } i \\ \bigcup_{p \in front(i)} i_{|p}.\sigma\big(\chi(i,p)\big) & \text{otherwise} \end{cases}$$

Deciding whether or not an interaction *i* accepts the empty trace is done via static analysis using an inductive reasoning on the expression denoting *i*. In short, only the empty interaction Ø or the repetition operators (when the content of the loop is repeated 0 times) can imply the acceptance of the empty trace.

From the definition of the χ function, it is easy to deduce a relation in the style of those used by Plotkin to define operational semantics of process algebras. We note $i \xrightarrow{a} i'$ if there exists a position *p* in *front(i)* verifying $\chi(i,p) = i'$ and $i_{|p} = a$.

From an initial interaction i_0, it is possible to compute all the pairs (i_1, a_1) verifying $i_0 \xrightarrow{a_1} i_1$. This process of computing follow-up interactions can be repeated recursively so that we obtain a tree which root is i_0 and represents its set of accepted traces. Traces expressed by i_0 can then be obtained via the concatenation of the actions a_j that are successively observed on any path of the tree starting from i_0. Thus, $a_1 \ldots a_n$ is an accepted trace of i_0 if for all $j < n$, we have $i_j \xrightarrow{a_{j+1}} i_{j+1}$ and ϵ is accepted by i_n. The interaction of Figure 1 has exactly 4 actions in its frontier. Figure 2 illustrates these 4 actions as well as their follow-up interactions.

2.3 On the execution of actions

The definition of the execution function χ involves a number of concepts introduced in [4]. In particular, the mechanism of pruning enters into play for handling weak sequencing. For example, let us consider the execution of $l_3!m_4$ in Figure 2. If $l_3!m_4$ is the first action to occur then, in order to respect the top to bottom order of the diagram (i.e. weak sequencing), neither $l_3!m_1$ nor $l_3!m_3$ can occur. Indeed, if those actions were to occur they would have done so before $l_3!m_4$. Hence, both actions must be eliminated from the follow-up interaction and this is possible because they are within alternatives and loops. This process of pruning is illustrated on Figure 3. The interaction expression with *coreg* as root in Figure 3 is simplified with two purposes: eliminate all traces with an action on lifeline l_3 and preserve all other accepted traces. We do this by transforming the interaction expression. As the sub-interaction *strict*($l_1!m_1$,*seq*($l_2?m_1$,$l_3?m_1$)) accepts a unique trace containing the action $l_3?m_1$, the first alternative of the *alt* operator is no longer allowed and the sub-interaction with *alt* as top operator is reduced to its second alternative, here Ø, which by definition accepts only the empty trace and consequently avoids lifeline

l_3. As can be seen from Figure 3, the process of pruning an interaction is a local transformation guided by the lifeline to be avoided.

3 RUNTIME VERIFICATION AGAINST INTERACTION MODELS

3.1 Trace analysis

The interaction-based V&V tool implements algorithms for analysing traces w.r.t. interactions. These analyses determine whether or not a behavior given as a trace is accepted. More generally, given an interaction model and a trace, a verdict of conformance can be produced.

The principle of the algorithm is to consider the first element a_1 of the trace t to be analysed (therefore of the form $a_1.t'$), to execute it in the reference interaction i_0 and to remove it from the trace. This allows us to start again with all the interactions i_1 verifying $i_0 \xrightarrow{a_1} i_1$ and the remaining trace t'.

If the original trace t of length n can be emptied, then it means that it is accepted by the original interaction i_0 iff ϵ is accepted by the last interaction model i_n. If this is not possible then this means that the behavior t deviates from i_0.

In general, there might be several occurrences of the same action in a given interaction. Hence, there may be several paths in the analysis graph, as described in [4] (see Figure 4 where $l_1!m_1$ occurs twice in the top interaction).

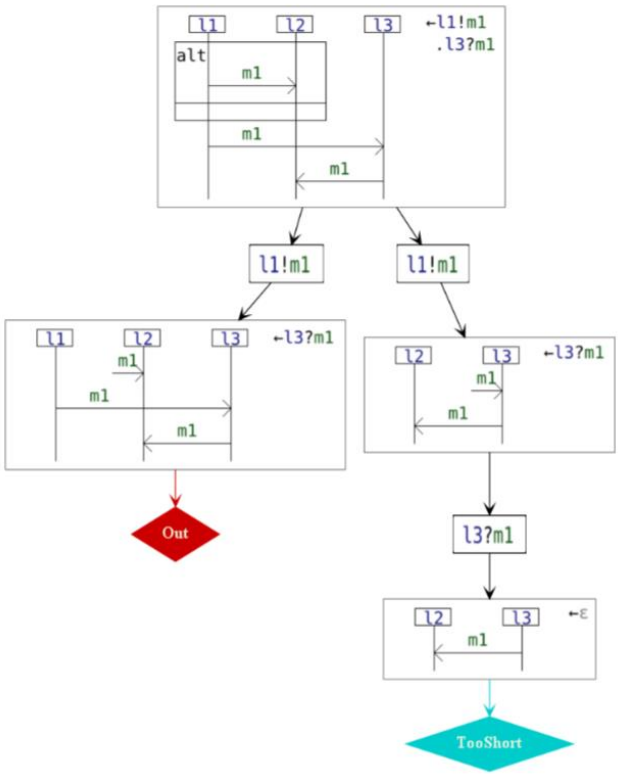


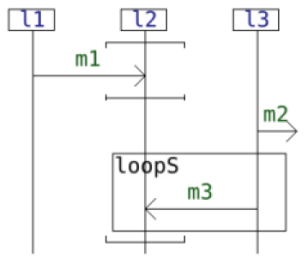
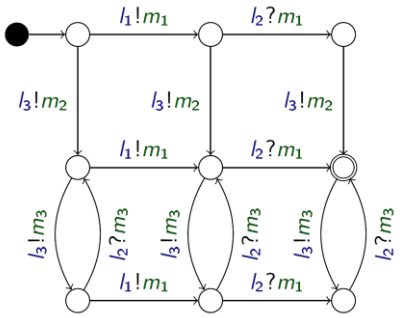
Figure 4 Trace analysis

The analysis process is illustrated on Figure 4 on the trace $t = l_1!m_1.l_3?m_1$. Here, the two manners to interpret the emission of m_1 yield two paths in the analysis graph. We provide a variety of local verdicts to decorate those paths. The *Out* verdict in the left branch signifies that it is not possible to interpret t as accepted if the first action $l_1!m_1$ is consumed considering the occurrence within the *alt* operator. It is not possible for l_3 to receive m_1 from the bottom left interaction. In contrast, if the first action $l_1!m_1$ is consumed considering the occurrence below the *alt* operator, it becomes possible to consume $l_3?m_1$. As the trace is now empty with an interaction that does not accept the empty trace, the *TooShort* local verdict identifies the trace under analysis as a strict prefix of an accepted behavior.

Trace analysis consists in exploring all branches of the execution tree, guided by the trace to be analysed, and annotating all these branches with a local verdict. Once these local verdicts are computed, it is then possible to produce a global verdict. The global verdict that is produced from Figure 4 is *WeakPass* because there exists a manner to execute the interaction that yields a *TooShort* local verdict. It would have been a *Pass* verdict if we had reached an interaction able to express the empty trace ϵ, and a *Fail* verdict if all paths led to *Out*.

A subclass of interaction models has regular semantics, meaning they can be associated with finite automata. This subclass uses exclusively strictly sequential loops (loopS). Table 1 depicts an example of such an interaction and one such finite automaton.

Table 1 Automata inference from interactions

Interaction	
Inferred finite automata	

Traces can be analyzed against automata generated from interactions. Such automata have the advantage of decoupling the exploration from the trace analysis, which can accelerate the analysis for divergent traces from the model as they require exploration of all paths of the exploration tree. Additionally, in case some actions are not observed (partial observability), the automaton can be used with a hiding mechanism to perform the analysis. The significance of the verdicts in such cases lies in their ability to infer the conformance of observed traces despite

the partial observability. While the analysis may not provide a complete picture due to the hidden actions, it still offers valuable insights into the observed behavior of the system under examination.

3.2 Multi-trace analysis

Whenever logs are collected, if actions can be ordered globally, whichever is the subsystem on which they occur, the execution scenario is characterized by a global trace. However, in all generality, as the subsystems of a DS can be distributed across distant machines, they do not share a common clock [5] and such a centralization and reordering of logging is not possible. Still, it may be so that groups of subsystems do share a common clock. We call those groups *co-localizations*. It is then possible to collect local traces, each representing a local order of actions on a co-localization. By gathering those local traces, we can then characterize the execution scenario as a *multi-trace* (a structured collection of local traces). In [4], we defined a multi-trace analysis algorithm for the fully decentralized case.

Multi-traces can be defined as follows. A multi-trace μ for a partition[1] C of the set L of lifelines is defined as a tuple of traces, each defined over actions occurring on a specific co-localization c from C. For any such c in C, we denote by $\mu_{|c}$ this local component of μ. We can define a projection from global traces to multi-traces.

This is illustrated on Figure 5 where, from a given initial trace, we propose two different projections: on the left, as a multi-trace [4] on the discrete partition of lifelines, and, on the right, on a certain partition $C = \{ \{l_1\}, \{ l_2 , l_3 \} \}$.

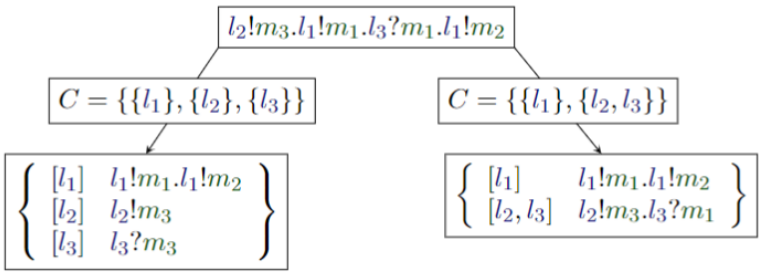


Figure 5 Projections onto multi-traces

Works [3] and [4] propose algorithms for verifying global traces and decentralized traces against interactions. The notion of co-localization generalizes and bridge the gap between those two algorithms. Indeed global traces are multi-traces on the trivial partition $C = \{L\}$ and multi-traces are multi-traces on the discrete partition.

4 CASE STUDY AND OFFLLINE RUNTIME VERIFICATION

[1] For a set X, we denote *Part(X)* the set of its partitions, i.e. the set of collections C of subsets of X verifying $\bigcup_{c \in C} c = X$ and $\forall (c, c') \in C, c \neq c' \Rightarrow c \cap c' = \emptyset$.

4.1 Case study : Handling certificate in vehicular communication

In V2V/V2I/V2X communication, Intelligent Transportation System (ITS) stations (encompassing both vehicle and roadside units), engage in the exchange of Cooperative Awareness Messages (CAM) to share vehicle status information such as position, speed, and direction. These communications can be the target of malicious attacks to either hampered the traffic or try to affect autonomous systems behavior and user security using such technics as injection of false information or the interception of messages to track vehicle locations and infer sensitive user profiles. For this, V2X messages and more particularly Secure CAMs, incorporate a specific field called "Secured Packet" housing cryptographic and certificate exchanged mechanism, also known as an Authorization Ticket (AT).

Within a CAM message, transmitting vehicles append a digital signature and a digital certificate. Upon reception, receiving vehicles utilize the public key within this digital certificate to validate the authenticity and integrity of the message by verifying the digital signature. Nevertheless, this robust security approach can lead to message overload due to its verbosity. Therefore, ITS stations are equipped with the flexibility to transmit either the complete certificate or a digest, which encapsulates a portion of the certificate's hash alongside the algorithm identity used for its computation.
Employing a digest instead of sending the full certificate in vehicular communication presents several advantages:

- Firstly, it significantly reduces data transmission, thereby enhancing bandwidth efficiency and mitigating communication overhead, particularly in high-traffic scenarios prone to network congestion.
- Secondly, it reduces validation overhead as a digest of an already encountered certificate. This means that instead of rechecking the entire certificate, receiving vehicles can use the digest to quickly find the necessary certificate information from their own database.

Standard bodies ETSI and IEEE have delineated specific certificate mechanisms within CAM messages, as detailed in [6] and [7] respectively. Our focus lies in modeling and analyzing CAM communication flows designed to adhere to these established standards, ensuring security and interoperability within V2X environments, key to the safety and efficiency of AI based autonomous vehicular solutions.

4.2 Modeling with interactions

In V2X communication, scenarios often concerns the movement of ITS stations, leading to situations with different combinations of stations entering and leaving communication ranges. These scenarios highlight the dynamic nature of network connectivity and require adaptable communication protocols to maintain secure data exchange. The modeling we present here focuses on situations where an ITS station, such as ITS_S0 (e.g., a car passenger), moves into the communication range of another station, ITS_S1 (e.g., another car passenger or a roadside unit), as depicted in Figure 6.

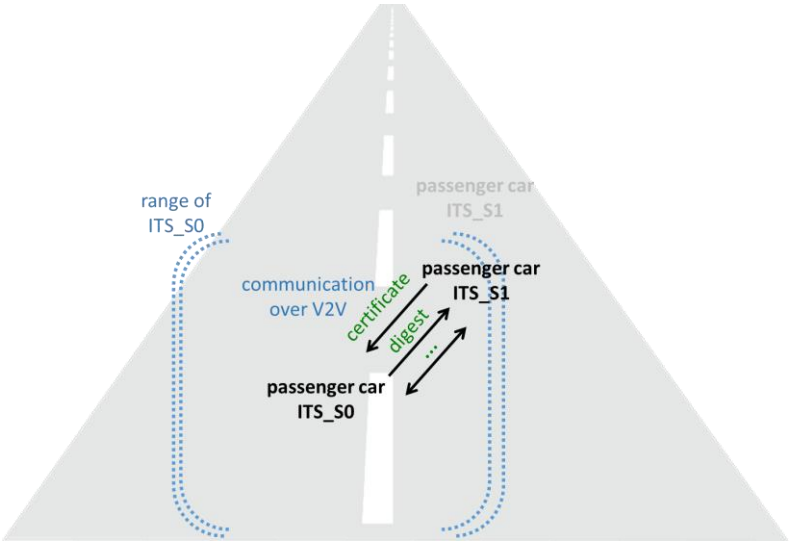


Figure 6 Enter range situation

In Figure 7, we present one possible intended interaction between ITS_S0 and ITS_S1 as follows. Some other variant behaviors are depicted in Figures 8, Figure 9 and Figure 10. The interaction depicted in Figure 7 unfolds from top to bottom as follows:

1. **Initial certificate exchange:**

At start, ITS_S0, operating in isolation, sends a certificate to establish its identity within the V2X communication environment, signaling its presence and readiness for communication with other ITS stations.

2. **Loop with digest transmission (ITS_S0 to its environment):**

While operating in isolation, ITS_S0 continuously sends digests to its environment, preparing for potential communication with other ITS stations entering within its communication range. According to following excerpt of [6], the behavior of an ITS station is to periodically send digests, and every 1s (here abstracted by a number X of sent digests), it sends (full) certificate:

" - As default, the choice digest shall be included.
-The choice certificate shall be included once, one second after the last inclusion of the choice certificate."

3. **ITS_S1 enters range - Certificate exchange between ITS_S1 to ITS_S0:**

As ITS_S1 enters the communication range of ITS_S0, ITS_S1 sends a certificate to ITS_S0, enabling ITS_S0 to recognize and authenticate ITS_S1. Here, certificate sent by ITS_S1 becomes known for ITS_S0.

In response, ITS_S0 sends its certificate (and not digest), following the requirements of [6] (REQ1):

"If the ITS-S receives a CAM signed by a previously unknown AT, it shall include the complete certificate immediately in its next CAM, instead of including the digest. In this case, the timer for the next inclusion of the complete certificate shall be restarted."

In fact, upon receiving a message from ITS_S1, ITS_S0 sends a complete certificate immediately instead of a digest as the received message is the first communication from ITS_S1 and thus the station was previously unknown for ITS_S0.

At this point, the certificate sent by ITS_S0 is known by ITS_S1 and the one sent by ITS_S1 becomes known to ITS_S0.

4. **Parallel Interaction:**

With both ITS_S0 and ITS_S1 aware of each other's full certificates, both stations can resume default behaviors. This can be modeled with two concurrent loops: a parallel loop with digests transmission and certificate exchange from ITS_S0 to ITS_S1, and another parallel loop with the same actions from ITS_S1 to ITS_S0.
It should be noted that the behavior may diverge from these interactions while remaining acceptable, particularly if ITS_S1 leaves the range of ITS_S0 (or vice versa).

Figures 8 models a variation of the previous interaction: ITS_S0 first sends its certificate upon ITS_S1 entering its range. This interaction is dual to the one given in Figures 7.

Figures 9 models another possible interaction where ITS_S1 receives a digest upon entering the range of ITS_S1. In this case, ITS_S1 sends its certificate following REQ1 (unknown certificate situation). Additionally, ITS_S1 is proactive and requests the unknown full certificate via including the request inlineP2psdRequest and this according to following requirement of [6] (REQ2):

“The component inlineP2pcdRequest shall be included and shall contain the digests of certificates currently unknown to the ITS-Station in the following cases:

if the ITS-S received a CAM with the component signer of SignedData set to the choice digest, and this digest points to an unknown authorization ticket; (…)”

Here, we distinguish a message label “certificate_ inlineP2psdRequest” from simple label “certificate” to abstract a CAM message including besides the full certificate, the request inlineP2psdRequest.

In response, ITS_S0 sends its certificate following the requirement of [6] (REQ3):

“If an ITS-S receives a CAM that includes a tbsdata.headerInfo component of type inlineP2pcdRequest, then the ITS-S shall evaluate the list of certificate digests included in that component. If the ITS-S finds a certificate digest of the currently used authorization ticket in that list, it shall include the choice certificate immediately in its next CAM, instead of including the choice digest.”

Figures 10 models a variation of the previous interaction: ITS_S1 first sends digest upon ITS_S1 entering its range. This interaction is dual to the one given in Figures 9.

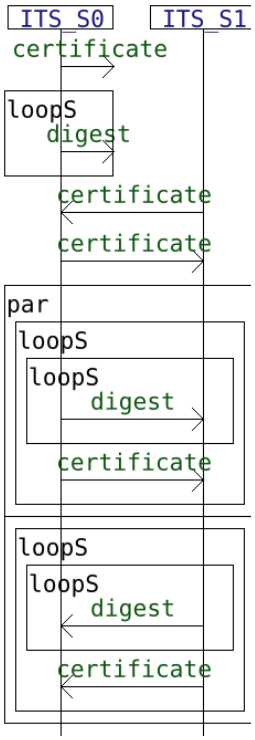


Figure 7 Enter range scenario 1

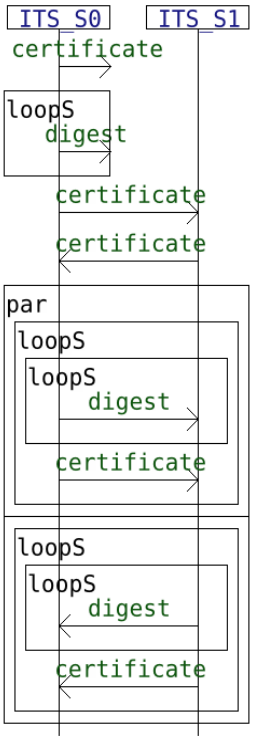


Figure 8 Enter range scenario 2

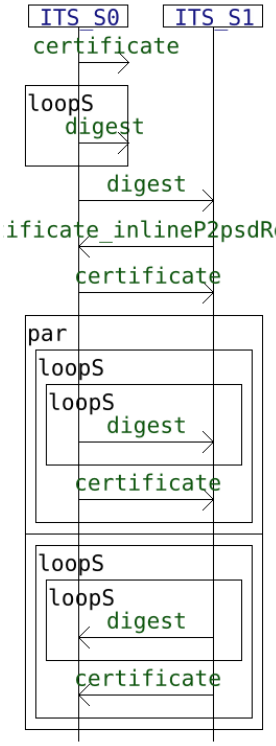


Figure 9 Enter range scenario 3

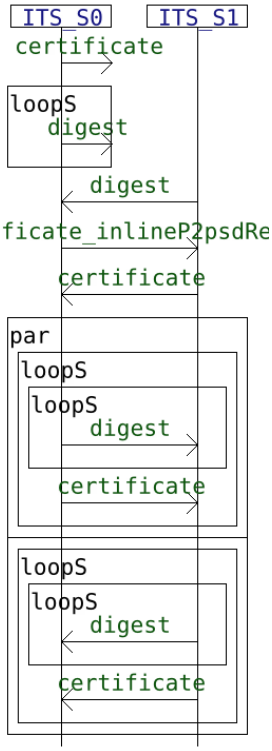


Figure 10 Enter range scenario 4

4.3 Runtime verification

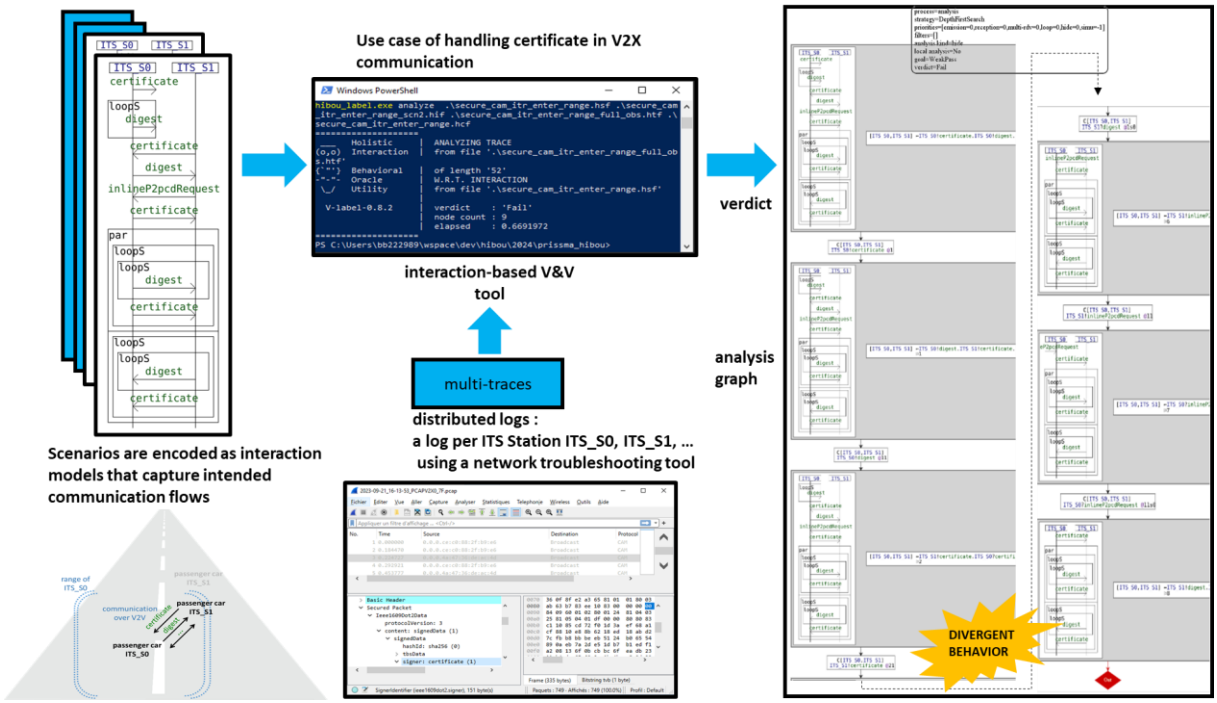


Figure 11 Offline runtime verification workflow

Figure 11 outlines the RV verification workflow. The Interaction-based V&V tool takes an interaction model and (multi-)trace as input, computing a verdict on its adherence to the interaction. In our case, we perform verification for each interaction model developed for our case study. If we encounter divergence from all models, then it raises a point of attention: either it indicates non-conformance to the requirements or suggests that our interaction model base is incomplete, necessitating enrichment of our modeling.

4.4 Multi-trace analysis

Let us consider the following multi-trace in the input format of the interaction-based V&V tool:

```
{
[ITS_S0]
        ITS_S0 ! certificate.
        ITS_S0 ! digest.
        ITS_S0 ? certificate.
        ITS_S0 ! certificate.
        ITS_S0 ! digest.
        ITS_S0 ? digest.
        ITS_S0 ? digest.
        ITS_S0 ! digest.
        ITS_S0 ! certificate.
        ITS_S0 ? certificate
        ;
[ITS_S1]
        ITS_S1 ! certificate.
        ITS_S1 ? certificate.
        ITS_S1 ! digest.
        ITS_S1 ? digest.
        ITS_S1 ! digest.
        ITS_S1 ? digest.
        ITS_S1 ? certificate.
        ITS_S1 ! certificate
}
```

The tool computes adherence to the first interaction of Figure 7.
The following multi-trace diverges from all developed interactions:

```
{
[ITS_S0]
        ITS_S0 ! certificate.
        ITS_S0 ! digest.
        ITS_S0 ? certificate.
        ITS_S0 ! digest.
        ITS_S0 ! digest.
        ITS_S0 ? digest.
        ITS_S0 ? digest.
        ITS_S0 ! digest.
```

```
        ITS_S0 ! certificate.
        ITS_S0 ? certificate
        ;
    [ITS_S1]
        ITS_S1 ! certificate.
        ITS_S1 ? digest.
        ITS_S1 ! digest.
        ITS_S1 ? digest.
        ITS_S1 ! digest.
        ITS_S1 ? digest.
        ITS_S1 ? certificate.
        ITS_S1 ! certificate
    }
```

Part of the analysis graph is given in Figure 12. The divergence is due to emission of a digest by ITS_S0 at some point in the multi-trace (see action highlighted in red) which cannot be matched against any follow-up action in the interaction of Figure 7. In such situation, ITS_S0 is expected to send full certificate as it received an unknown certificate in previous step, which does not conform to REQ1.

Figure 12 Analysis graph of a multi-trace analysis against interaction of Figure 7

4.5 Analysis under partial observability

The initial objective of the current deliverable was to evaluate PRISSMA use-cases implementations. However, the required CAM captures for this validation demonstration that were to be generated during the other project tests (as in task 5.6), were not produced in the context of those tests. The tests performed in the other project tasks failed to capture proper secured CAM exchanged. Because either these features were not implemented by the tested equipment's or not managed to be configurable within the project constraints and resources. The captured data used were provide by partners who generated them during the IRT SystemX project TAM (https://www.irt-systemx/projets/tam).

The communication captures (CAM) that we had at our disposal correspond to the "Enter range" situation of real V2V (Vehicle-to-Vehicle) communications. These captures represent actual data collected during real-world scenarios using the Wireshark tool at the ITS_S0 station level. Examples of captures from various executions are depicted in Figure 13, Figure 14, and Figure 15. In these figures, the involved stations are car passengers identified by their IDs 1234 (ITS_S0) and 1235 (ITS_S1).

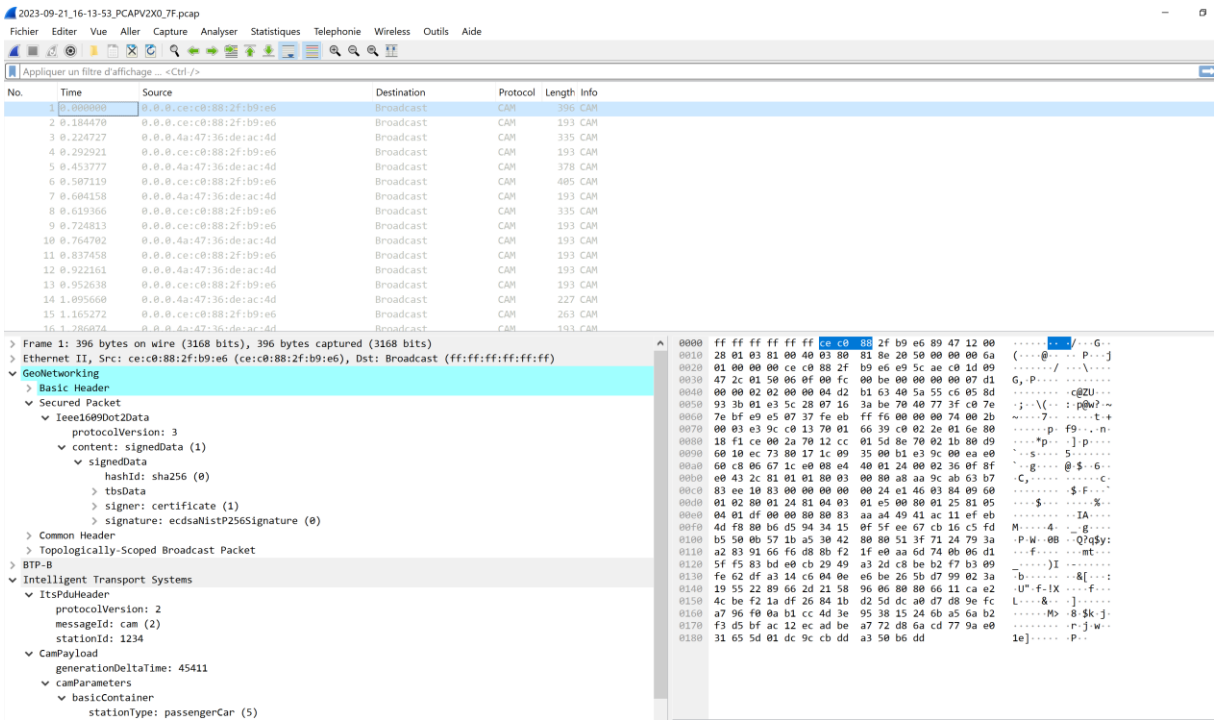


Figure 13 CAM message example 1: Sending certificate

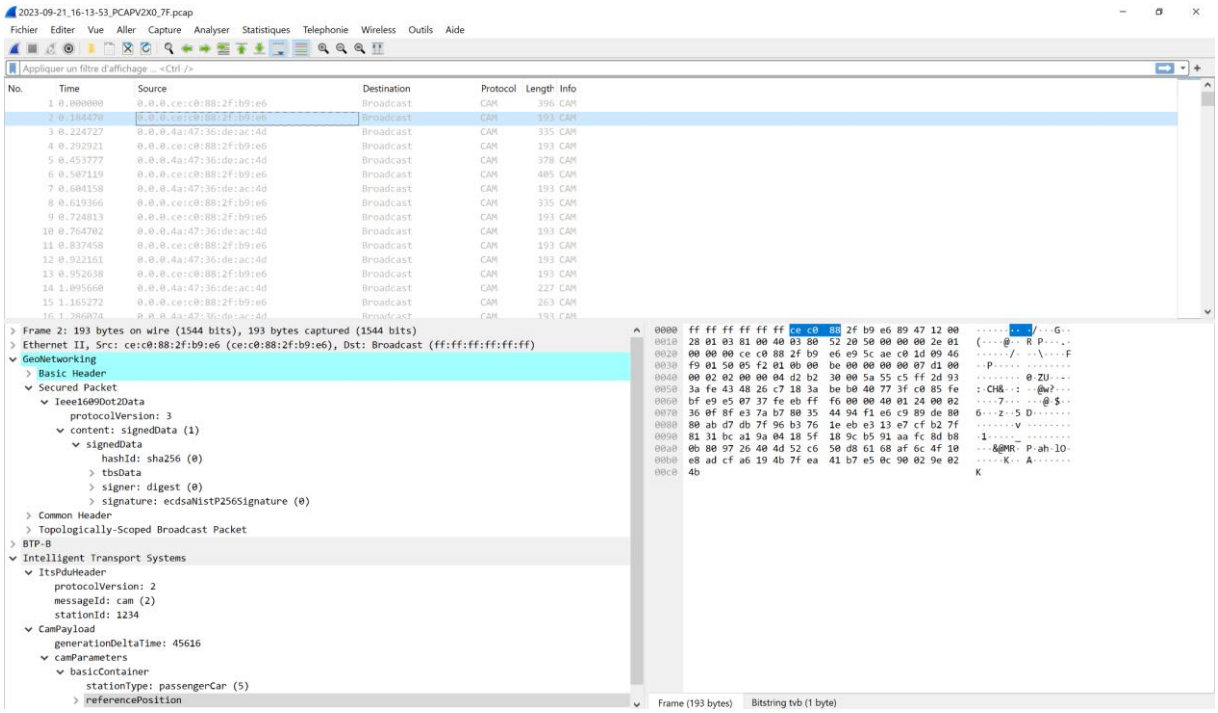


Figure 14 CAM message example 2: Sending digest

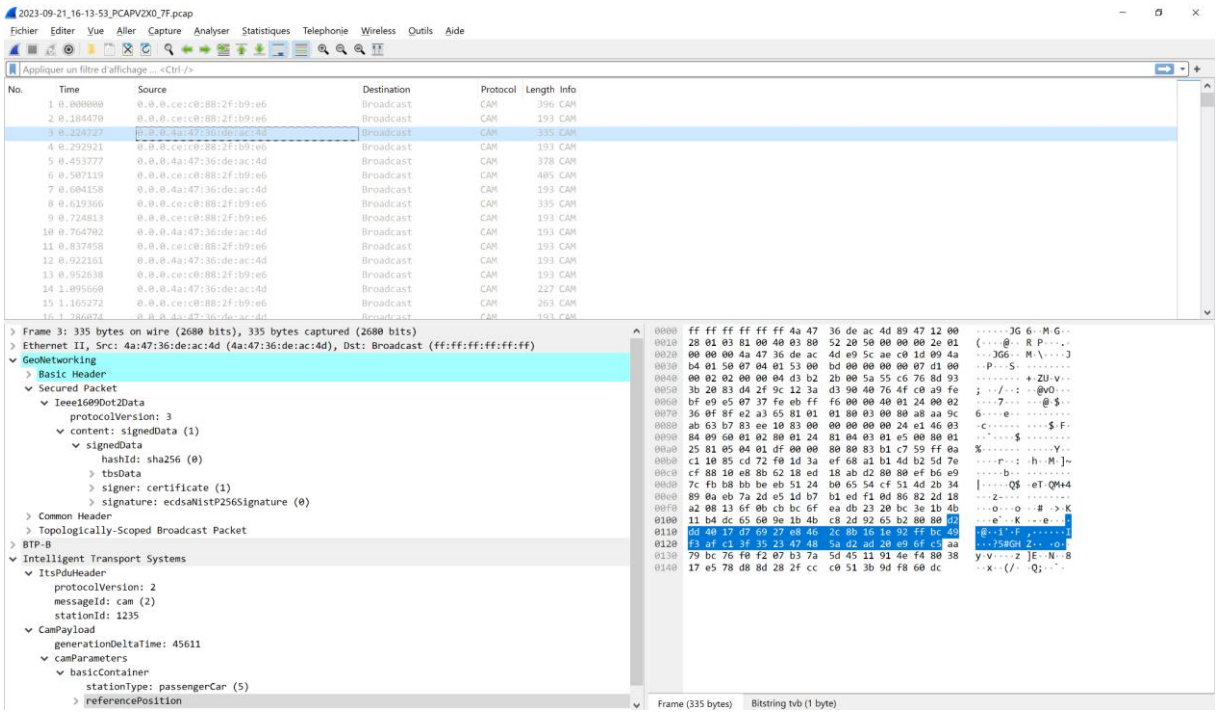


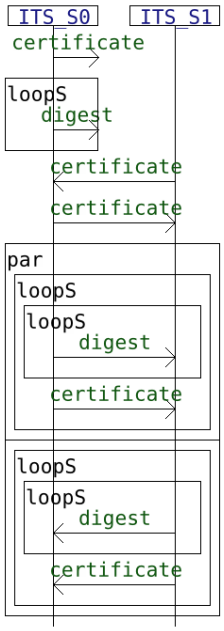
Figure 15 CAM message example 3: Receiving certificate

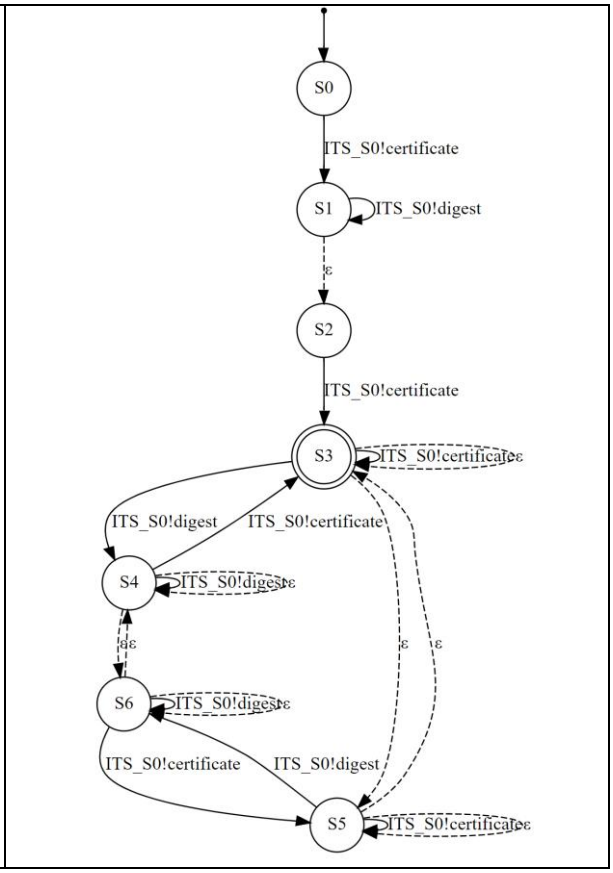
The captures have been collected at the ITS_S0 station level, allowing us to identify messages emitted by ITS_S0 (Figure 13 and Figure 14) and those received from ITS_S1 (Figure 15).

However, the reception of messages is not recorded at the moment they are considered by ITS_S0, but rather when they are observed entering the input port of the station. Consequently, we run the analysis under partial observation concerning the developed interactions by removing the direct receptions of messages and only considering the emissions using the finite automata-based trace analysis method. An example automaton used in the trace analysis is depicted in Table 2. For instance, in the example automaton, the epsilon (ϵ) transitions are used to represent unobservable actions. Tools like Wireshark allow us to observe message receptions when they arrive at some port, not when they are actually processed by the ITS stations.

To perform analysis with this information, we need more advanced mechanisms that account for the fact that some messages are delayed (typically queued) before being processed.

Table 2 Reference automata for analysis under partial observability

Interaction Figure 7	

Automata are obtained by projecting onto ITS_S0 and then hiding receptions by ITS_S0 (unobserved actions).	

The analysis for the recorded traces did not reveal any non-conformance and thus allowed us to formally validate the implementation of the modules in this operational scenario for the different requirements defined in section 4.2.

5 CONCLUSION

Deliverable 5.5 provides a tooled V&V methodology leveraging interaction models like UML Sequence Diagrams and Message Sequence Charts. This interaction formalism is particularly suitable for distributed systems and communication protocols, which are predominantly involved in connected autonomous vehicular systems. Equipped with semantics, the considered interaction formalism enables rigorous runtime verification to reason about the operational behaviors of the involved subsystems and detect divergences from the interaction models.

We have illustrated this methodology with a case study of V2V/V2I/V2X secured communication based on ETSI's Cooperative Awareness Messages (CAM). The framework can be applied in scenarios of partial observability, where some subsystems or actions are not observed due to the unavailability of observation points or lack of instrumentation. Integration with widely used network debugging tools like Wireshark is also enabled, ensuring compatibility with existing practices and facilitating the troubleshooting and analysis of captured communication flows.

This case study demonstrates that formal validation can be used in the validation of very complex AI based distributed systems where high level of assurance are required and where current human based validation are very prone to errors. If modeling efforts are still potentially high the benefits of formal and thus exhaustive demonstration that implementations meet explicitly defined security requirements is a major contribution to the validation of AI based autonomous transport systems. Also, it can be noted that due to their technology agnostic modelling process,

the same models can actually be reused by any automotive actor to validate their implementations offline or in operational environments. Thus, not requiring redefining models for every evaluation, the developer of the Target Of Evaluation (TOE) only needing to implement plugins to transform operational or tests data in the proper V&V inputs format and not to possess formal modelling expertise.

REFERENCES

References

[1] OMG, «Unified Modeling Language,» 2017. [En ligne]. Available: omg.org/spec/UML/.

[2] ITU, «Message Sequence Chart (MSC),» 2011. [En ligne]. Available: itu.int/rec/T-REC-Z.120.

[3] Erwan Mahe, Christophe Gaston, Pascale Le Gall, «Denotational and operational semantics for interaction languages: Application to trace analysis,» *Sci. Comput. Program*, 2024.

[4] Erwan Mahe, Boutheina Bannour, Christophe Gaston, Arnault Lapitre, Pascale Le Gall, «A small-step approach to multi-trace checking against interactions,» *SAC*, 2021.

[5] Lamport, Leslie, «Time, clocks, and the ordering of events in a distributed system,» *Concurrency: the Works of Leslie Lamport*, 2019.

[6] (ETSI), European Telecommunication Standard Institute, *ETSI TS 103 097 Intelligent Transport Systems (ITS); Security; Security header and certificate formats;*, 2021.

[7] Society, IEEE Vehicular Technology, *IEEE Std 1609.2 IEEE Standard for Wireless Access in Vehicular Environments - Security Services for Applications and Management Messages*, 2016.