Virtual Verification of E/E Architectures for Secure Automated Driving Functions

Kevin Neubauer*, Marcel Rumez†, Huiying Tremmel†, Augusto Hoppe*,
Reiner Kriesten†, Philipp Nenninger†, Eric Sax*, and Jürgen Becker*
*Institute for Information Processing Technologies (ITIV), Karlsruhe Institute of Technology (KIT), Germany
Email: {firstname.lastname}@kit.edu
†Institute of Energy Efficient Mobility (IEEM), Karlsruhe University of Applied Sciences (HKA), Germany
Email: {firstname.lastname}@h-ka.de

2021 IEEE International Symposium on Systems Engineering (ISSE) | 978-1-6654-3168-2/21/$31.00 ©2021 IEEE | DOI: 10.1109/ISSE51541.2021.9582552

***Abstract*—Automotive trends, such as autonomous and connected driving, are increasing the number of sensors to percept the environment precisely. To support automated driving functions, vehicles share information via wireless interfaces (e.g., cellular) with Other Road Users (ORUs). This growing degree of connectivity also increases the risk of cyberattacks. According to the United Nations Economic Commission for Europe (UNECE) regulation authority (WP.29), the integration of security countermeasures (e.g., Intrusion Detection System (IDS)) is mandatory for Original Equipment Manufacturers (OEMs). To verify the effectiveness of such security measures for Advanced Driver Assistance System (ADAS) functions, aspects such as vehicle behavior, network communication and the environment have to be taken into account. The use of model-based simulation approaches enables an early virtual verification of electrical/electronic architectures (E/E architectures) before the associated hardware is completely developed. In this paper, we describe a multi-domain co-simulation approach for a scenario-driven and reactive testing of E/E architectures that form with their corresponding environment cyber-physical systems (CPSs). It is based on deriving simulation models from the established E/E architectures modeling tool PREEvision. Moreover, we show the development of an IDS by applying our simulation approach in a model-based manner and evaluate it with a pre-defined attack scenario. As a conclusion, we discuss how this approach can support identifying design flaws for developed countermeasures and help to improve or establish standards for model-based security testing of E/E architectures.**

***Index Terms*—Security, E/E Architecture, Simulation, Virtual Verification**

I. Introduction

The automotive industry is currently undergoing a transformation, driven by trends such as automated and connected driving functions, electric mobility and changing customer requirements [1]. With the launch of level 3 of the SAE J3016 levels [2] of driving automation, new driving features will hit the road. The latest launch predictions are based on the years 2021 to 2024 [3]. This upgrade will result in a shift in responsibility between the driver and control system (vehicle). As a result, in addition to control tasks (steering, acceleration, deceleration), the system must monitor the complete environment even more than a real driver today. One of these level 3 Advanced Driver Assistance System (ADAS) is the Automatically Commanded Steering Function (ACSF), which is specified by the United Nations Economic Commission for Europe (UNECE) [4]. This function, once activated by the driver, is able to independently perform a lane change maneuver while observing other vehicles (e.g., by determining distance using radar or cameras) on a highway. For testing such kind of ADAS functions, it is no longer sufficient to verify corresponding hard- and software modules (e.g., Electronic Control Units (ECUs) or software function) separately in a bottom up manner based on different test stages (unit, integration, system and acceptance) of the V-model [5]. The reason is, that ADAS functions not only implemented with on-board components, but detect and interact with the environment. As a result, such functions can only be tested comprehensively late at the *system test* stage with high testing efforts due to real test drives. However, simulation can be used to realize these system tests earlier with Vehicle-in-the-Loop (ViL) [5] concepts. To close the loop between environment and vehicle, real components are represented by models in the simulation. The application of such tools enables more variability, observability and reproducibility in comparison to real test drives [5].

In addition, ADAS have to be protected against cyberattacks to ensure information security by integrating countermeasures, since attacks on vehicles have been increasingly published in recent years [6]. In this area, simulations also offer comprehensive ways for developers to test security measures against attacks (e.g., manipulation of sensor values) and identify possible impacts on the vehicle's behavior through graphical animations. In this paper, we address the following issues:

Problem: ADAS interact closely with the environment of the vehicle. For testing purposes, it is not sufficient to verify Electric/Electronic (E/E) domains separated in early development phases or the entire system at the end of the development.

Solution: Co-simulations consisting of electrical/electronic architectures (E/E architectures), coupled with the environment and a model-based, executable test-scenario in a reactive manner, allow ADAS to be tested extensively in terms of security aspects.

Contributions: 1) We present a generic, hardware-centric and integrated multi-domain simulation test environment based on a combination of the tools PREEvision, Ptolemy 2 (PT2) and OpenDS. 2) We show opportunities and drawbacks of a

model-based test derived from United Nations Economic Commission for Europe (UN/ECE) Regulation (UNR) according to security aspects. 3) We evaluate the approach based on an exemplary use case: the development and testing of an IDS for an ACSF based on cyber and physical detection features. Such kind of security measure is still under-investigated [7].

II. Background

A. E/E Architectures

An E/E architecture is an embedded system, where the electrical part includes the components for power supply, distribution, conversion and physical communication, such as wires or passive actuators and sensors with linear characteristics. The electronic part includes components for controlling the electrical current through circuits involving non-linear components, such as semiconductors. Usually, vehicle functions are implemented in software executed on microcontrollers. The structure of an E/E architecture can be described by a layered model, where every layer has a specific view described by block diagrams [8].

B. Cyber-Physical Systems

An E/E architecture forms together with the vehicle, in which it is embedded, a mechatronic system. If there is also a connection to other systems or the internet via a communication system (e.g., cloud), it is referred to as a cyber-physical system (CPS) (see Figure 1). Besides information flow (logical data) we differentiate in the modeling between material (e.g., air) and energy (e.g., current) flow [9], [10].

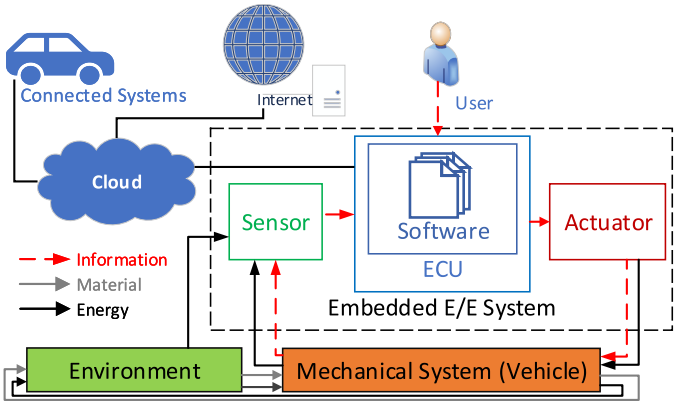

Figure 1. Adapted representation of a CPS according to [9], [10].

C. Information Security

Information security aims to protect information from unauthorized access to data, modification or other unintended actions, which can be mapped to the three main security objectives (confidentiality, integrity and availability) [11]. Countermeasures are used to ensure these objectives by mitigating the potential risk or in case of successful access to reduce the impact of such incidents. A risk comprises the probability, that an attacker or other threat source exploits a vulnerability (e.g., weak password). There are three fundamental classes of countermeasures (administrative, technical and physical). The first one includes for example security documentation or staff training. Technical measures involve firewalls, IDS or encryption techniques. The third class represent locked server rooms or locked internal and external doors of a building. Technical measures can be further classified into preventive and proactive. On the one hand, a firewall aims to control network packets based on pre-defined filtering rules to reduce the potential risk in a preventive manner. On the other hand, an IDS enables to detect attacks or attack attempts during their occurrence by analyzing network traffic or host system behavior, if preventive measures are bypassed or defeated. Regarding the location of their deployment, IDS can be classified into either host-based or network-based IDS. Host-based IDS (HIDS) analyze events and user behavior with regard to the granularity of individual devices. Network-based IDS (NIDS) can protect multiple devices or even entire networks at the same time, as they only monitor network traffic [12].

D. Tools

In this section we introduce the tools used in section V for the prototypical implementation of our approach.

PT2 is a multi-domain simulator, where different computational models, called directors, can be combined hierarchically. Blocks, called actors, are connected via relations between their ports in a graphical manner in order to communicate with each other by sending tokens [13]. We use it as a main simulator and for the time synchronization in a co-simulation.

OpenDS is an open source driving simulator written in Java, which was mostly developed at the German Research Center for Artificial Intelligence (DFKI). The core components are vehicle dynamics and a traffic simulation, supporting the OpenDrive standard [14]. We use it for the co-simulation of vehicle dynamics and the environment.

PREEvision is a tool from the Vector Informatik GmbH [15] for the model-based development of E/E architectures (refer to II-A). The layers of an E/E architecture are organized as follows. On the *requirements layer* we find the documentation and management of requirements and customer features. They can be transferred into an implementation-independent concept within the *Logical Architecture (LA)* layer, consisting of sensor, function and actuator blocks that are connected through input- and output ports and represent an input, processing, output principle. Figure 2 shows an exemplary representation of this layer and following ones. The LA components can be implemented in both hardware and software. In the AUTomotive Open System ARchitecture (AUTOSAR)-compatible *Software Architecture (SA)* Software Component (SWC) blocks and their interactions are modeled, while in the *Hardware Architecture (HA)* the modeling of E/E component blocks, such as ECUs, sensors, actuator or gateways, their power supply as well as their communication takes place. Furthermore, detailed component models as well as the geometry design for locating the components are supported among other features.

III. Related Work

A. Verification of secure E/E architectures using Simulation

In the E/E architecture development, simulation is crucial to support E/E engineers in making the right design decisions as early as possible. This includes, for example, the identification of vulnerabilities at system and component level or the verification of a modeled logic. In [16], [17] executable simulation models are synthesized from static PREEvision E/E architecture models. Starting from the LA, the structure of which primarily forms the basis of the simulation model, the author assigns corresponding behavioral models to the static artifacts to get the dynamic logic for the simulation. This approach was supplemented in [18] so that vehicle dynamics and the environment are taken into account. In addition, from UNR derived executable test scenarios were used to optimize a logical vehicle function, but the implementation for the assessment of security mechanisms has not yet been done. Furthermore, the Eclipse Working Group „openPASS“provides a comprehensive framework to use virtual simulation, mainly for the assessment of safety impacts [19]. Since it is still a relatively young project, it is unclear how security aspects can be taken into account and evaluated in the future. For now, the architecture description required for large parts of a dynamic threat analysis (security assessment) is missing in this concept, although it would be needed. Furthermore, Kuhn describes the use of virtual prototypes for the assessment of advanced security mechanisms, as simple security mechanisms, such as security by obscurity, are no longer sufficient [20]. However, a static thread analysis and the dynamic analysis of the integrated security concepts are separated in this approach. It also lacks the consideration of the environment and standardized test metrics.

B. Automotive Security Measures

Due to the increasing number of attacks on vehicles, researchers have published various approaches to secure automotive E/E architectures. An overview of such attacks and countermeasures is given in [21], [22]. For the detection of such attacks, significant research efforts have been invested on the development of IDS approaches [23], [24], [25]. Lokman et al. [24] presented a taxonomy for Controller Area Network (CAN) bus based IDS including four main classes (deployment strategy, detection approach, attacking techniques and technical challenges). The detection approach class is further subdivided into signature-, anomaly-, and specification-based approaches. *Signature-based* techniques use already known attack patterns stored in the IDS and compare them with current activities on network or host systems. The drawback is that the recognition is limited to the known patterns and new attacks remain undetected. *Anomaly-based* methods analyze the system or network behavior in real time and compare it with a known normal behavior and identify anomalies based on deviations from a defined threshold. In order to record normal behavior, techniques based on statistics, frequencies and machine learning have been widely established. *Specification-based* approaches use the advantage of statically defined communication in vehicles. During development, the entire network properties are specified, such as the used message IDs, assignment of sender and receiver, definition of the payload per message and fixed cycle times for transmission. An anomaly will be detected, if analyzed network activities show a deviation from specified properties.

Automotive IDS approaches can be further classified by the type of detection features (cyber and physical) used according to Al-Jarrah et al. [7]. *Cyber features* represent characteristics of the protocol or communication data (e.g., cycle time of CAN messages). In contrast, *physical features* represent sensor data or vehicle state information derived from them (e.g., wheel speed, vehicle in motion). The vast number of automotive IDS approaches presented are based on using cyber features. Al-Jarrah et al. [7] reviewed 42 published IDS approaches and outlined that 81% based on cyber-features, 5% used physical-features and 10% both features as a hybrid system (4% of these could not be classified). Furthermore, Loukas et al. [26] published a hybrid IDS based on deep learning by using cyber and physical features (e.g., network traffic rates, CPU utilization, encoder data from wheel motors or power consumption). Another hybrid IDS was presented by Weber et al. [27], which uses statically specified communication properties (e.g., cycle time or payload length of CAN messages) as cyber features. In addition, physical signal features (e.g., excessive slope of a sensor signal) are verified regarding their plausibility by using Artificial Intelligence (AI). To the best of our knowledge, there is no IDS approach which use either physical or a combination of cyber and physical features to secure ADAS functions. Therefore, we present a hybrid IDS approach as a use case for a simulation environment to evaluate security measures of E/E architectures with focus on ADAS functions.

C. Security Testing of ADAS Functions

To verify implemented security countermeasures according to requirements, static and dynamic security test methods (e.g., static application security testing (SAST), fuzz- and penetration testing) are performed at different levels on the right branch of the V-Model [28]. Furthermore, there are active research efforts addressing the area of model-based security testing in recent years through various published approaches, which Felderer et al. classify with a taxonomy [29]. The researchers Appel et al. also explain in [30] the necessity of early model-based testing with regard to security already during the design phase to verify implemented security mechanisms. They also explain, that the required test environment has to be capable of simulating vehicle models, environment and network communication to cover all aspects of ADAS functions. Moreover, they provide a test bed consisting of different software tools (e.g., MATLAB & Simulink, CARLA, ROS) and real hardware components with CAN interface. However, the test bed is limited with respect to the variation of the underlying E/E architecture. The test bed does not provide the ability to simulate different architecture designs (variants)

with associated network technologies since the architecture is rigidly coupled to the hardware used. Therefore, only a limited number of attack vectors can be mapped in terms of network paths.

IV. Multi-Domain Modeling & Simulation Concept

A. Requirements for the Simulation Environment

The simulation environment should cover the vehicle model, network and environment aspects to test security mechanisms for ADAS functions comprehensively. If a function developer executes different test cases (e.g., manipulation of sensor values), the corresponding vehicle behavior can be directly analyzed through a graphical visualization. In comparison to simulate one aspect standalone, it is challenging to identify resulting impacts of executed test cases. The simulation environment should also be scalable, i.e. allow the addition of further hardware or software components (refer to II-D), so that either all parts relevant to a use case or the entire E/E architecture can be considered. It should be also noted that all parts of the target E/E architecture model must be available. An Original Equipment Manufacturer (OEM) does not develop an E/E architecture from scratch for every vehicle model, typically. It is much more an evolutionary process, in which appropriate adjustments are made for product lines or variants depending on the requirements [31]. This means that when having to make a design or technology decision, an architect normally can rely on an existing architecture and its variants. Here adjustments are applied and can be evaluated by comparing different realization options or by using a suitable and reactive testbench that covers executable test scenarios and evaluation criteria derived from the requirements or accreditation criteria. The simulation environment should cover this.

B. Deriving Simulation Models from E/E architecture Models

We base our approach on [17] where static PREEvision E/E architecture models are used to synthesize executable simulation models (refer to III-A). However, in our approach we do not use the structure of the LA component model as the basis for synthesis, but rather the HA in order to be able to carry out a dynamic threat analysis. So, the hardware components, their interface and their communication build the top-level structure of the simulation model. It allows us to define concrete attack paths or targets in order to evaluate the corresponding defense mechanisms in the appropriate context.

The behavior can be modelled in PREEvision in the form of state-charts an annotated to the block prototypes on the LA, SA and HA. For profound modeling, PT2 proxy blocks can be used on a prototypical Behavioral Logical Architecture (BLA), described in [17]. However, this implementation is not sufficient for our hardware-centric synthesis, yet. Therefore, we use the information available in a PREEvision E/E architecture model and design manually the PT2 simulation model, which forms the basis for later automation.

Figure 2 shows the top-level concept for deriving simulation models from PREEvision E/E architecture models at the highest level of abstraction. In the upper part, we see the layers of an PREEvision E/E architecture model relevant to our approach. In the lower part, the top-level structure of the target PT2 multi-domain simulation model is shown. We can see that the main structure of the simulation model is based on the structure of the HA. In addition, details from other layers are added by propagating the mappings.

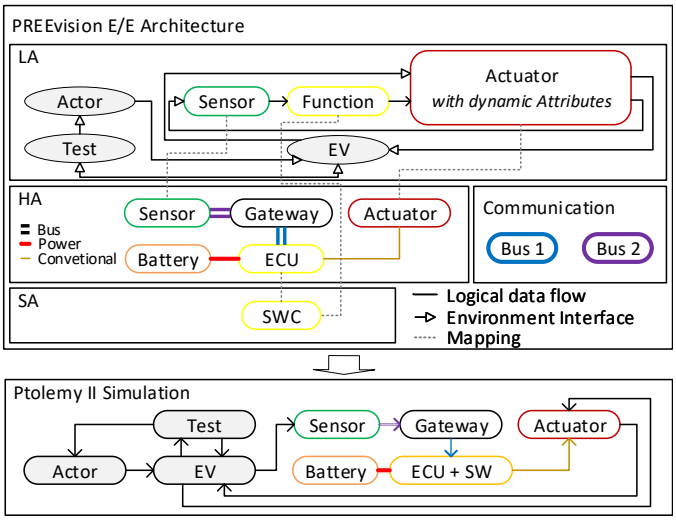

Figure 2. Concept for deriving hardware-centric and multi-domain PT2 simulation models from different PREEvision E/E architecture layers.

SWCs, defined in the SA, are mapped to ECUs. In the simulation model, they are encapsulated by the ECU and thus form a part of them. Together with the modeled communication, the signal paths between the components are integrated as well.

Finally, environment components (gray filled ellipsis) and interfaces to it, needed to implement a reactive test bed, can be found in the LA. Here, we must consider that the E/E architecture contains parts that are coupled with each other or with the environment. An actuator can have dynamic attributes that change over time, such as the rotational speed of a motor. If such an attribute is provided by a sensor to the E/E architecture, we have an indirect coupling between the sensor and the actuator over the environment. In addition, sensor signals may depend on *Actors*, such as traffic lights or Other Road Users (ORUs), that are part of the environment and interact with the Ego Vehicle (EV). In this case the logical sensor-values are provided to the E/E architecture through the *EV*. Moreover, the EV encapsulates and provides the vehicle dynamic forces, vehicle attributes and states, such as the current speed. As the EV receives control signals from the E/E architecture and provides logical sensor data to it, a closed loop between the E/E architecture and the (physical) environment is formed (refer to II-B). In this way, the EV forms an interface between the E/E architecture and the environment that can be also used for a co-simulation [18] (see IV-C). The environment component *Test* encapsulates a corresponding and reactive test metric, providing logical signals to the EV. It monitors all states and signals of the

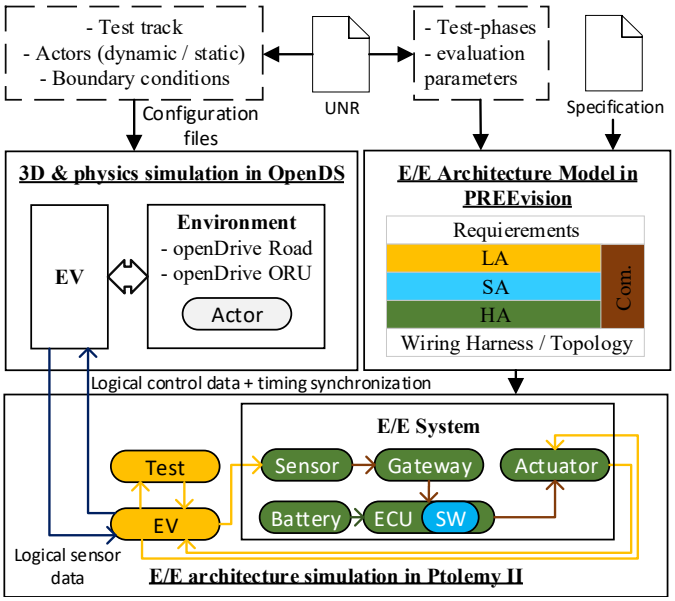

Figure 3. Modeling process and co-simulation framework, adapted from [18]

EV that are relevant for testing. Optionally, monitoring states and signals of the E/E components is also possible.

C. Co-Simulation Framework

Figure 3 shows our co-simulation framework. It is based on [18], but adapted to implement a hardware-centric simulation that comprises additionally the HA and the SA. Beginning from the top of the Figure, a UNR is used to derive specific test scenarios (see section V-B2) as well as the configuration for the environment including its actors. A PT2 simulation model is derived from an PREEvision E/E architecture as described in section IV-B. It includes, besides the structure of the E/E system, the complemented behavior of the E/E components as refinements and particularly a UNR test scenario in the form of a state machine within the environment component *Test*. The block EV serves as a proxy of an EV modeled and simulated in the 3D and physics simulation in OpenDS. In this way, the PT2 EV forms the co-simulation interface, encapsulates the timing synchronization and provides logical signals in both directions, where control commands come from the E/E system via actuators and sensor signals (coming from the OpenDS EV) are provided to the E/E system through sensors. Thereby, a signal conversion takes place in the sensor and actuator blocks in order to map hardware-specific characteristics to the logical signals.

V. Prototypical Implementation of an IDS

A. Misuse Case and Test Procedure

For the model-based and security-driven development of the IDS, we assume that a virtual attacker has gained control over the E/E system via remote access. Such attacks have been successfully demonstrated [32]. Thus, the attacker is able to manipulate specific internal sensor signals. The attack will impact the correct operation of an ACSF of category C [4]. We derive our test scenario from [4] and [33]. The procedure for the evaluation of the IDS is depicted in Figure 4.

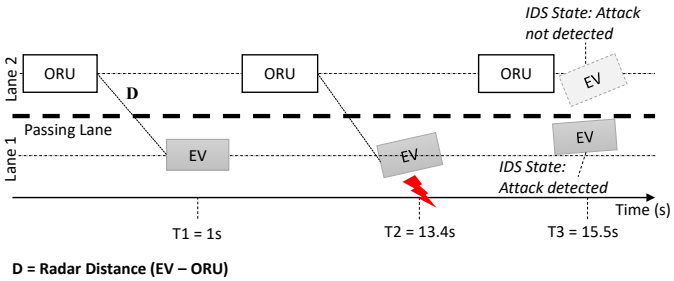

Figure 4. c structure of the specified test scenario for the evaluation of the IDS.

The simulation involves two vehicles (EV, ORU) moving autonomously with activated Lane Keeping Assistant (LKA) and Adaptive Cruise Control (ACC) on different lanes. At the start of the simulation (T1), the EV accelerates from 0 km/h to 80 ± 2 km/h, while the ORU accelerates from 0 km/h to 120 ± 2 km/h. They have an initial distance of 300 m. At T2 both of them have reached their constant target velocities and a critical distance $S_{critical}$ between 60 and 62 m. If the distance is less than $S_{critical}$, a collision would occur when the EV changes the lane. At this point we examine two variants of a misuse by an attacker. 1) The logical value of the radar distance, used by the ACSF, is manipulated by the attacker. As a result, the outgoing sensor value remains constant at D = 60 m using a limiter (refer to Figure 7. The IDS uses a redundant Car-2-X signal for the detection. 2) Additionally, to the first misuse, the redundant Car-2-X signal gets delayed in time.

B. Realization in Ptolemy II

The top-level simulation model in PT2 is shown in Figure 5. In line with the framework depicted in Figure 3, we find the block *EgoCar* (EV) as the interface to OpenDS with its input and output signals, as well as the environment component *Scenario* (Test). The internal state machine of the latter is shown and explained in V-B2. The E/E system model consists of *sensors* (green), *actuators* (red), *ECUs* (blue) and an *Attacked Gateway* (purple). Here the radar signal, representing the distance to the ORU, is manipulated by the attacker. *Logical signals* are marked in yellow, *bus signals* in green, *conventional signals* in gray and the *attacked bus signal* in red. The *ADAS High Performance Computer (HPC)* runs the IDS, the ACSF and ADAS driving functions. These are LKA and ACC for an autonomous lateral and longitudinal steering.

1) IDS Detection Algorithm Implementation: For detecting anomalies of the radar sensor information D (distance between EV and ORU), our approach is based on physical features, information redundancy and monitoring. In detail, we use the derivation of the radar distance with respect to time $D' = \Delta v_{radar}$ and the delta speed information of both vehicles:

$$\Delta v_{car2x} = v_{ORU} - v_{EV} \quad (1)$$

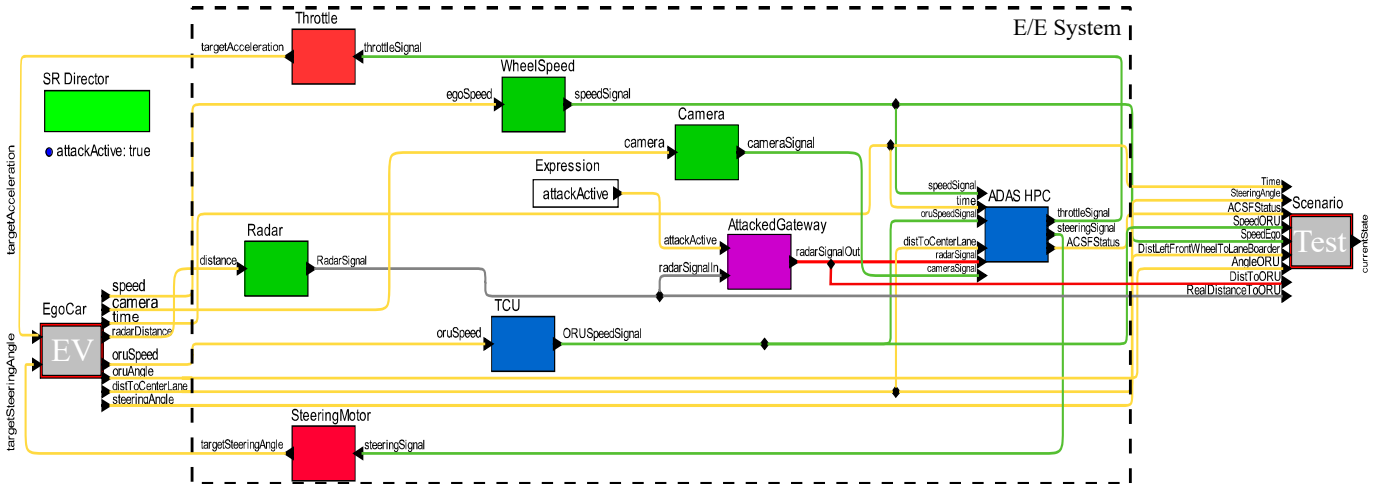

Figure 5. Ptolemy II top-level model of our reactive test bench, comprising the E/E System under test, the environment component EV and the environment component "Test" that includes the test scenario.

We assume that the speed information of the ORU is transmitted via Car-2-X communication (received through a Telematic Control Unit (TCU)). Using a redundant signal source minimizes the risk of this being manipulated by the attacker at the same time. Based on the two different speed signals we calculate again the delta of both signals:

$$\Delta v_{radar,car2x} = \Delta v_{radar} - \Delta v_{car2x} \tag{2}$$

The signal $\Delta v_{radar,car2x}$ forms the foundation for our IDS detection method. We analyze the signal with respect to the rate of change by calculating the moving average ma with a width of 2s respectively M = 20.

$$ma = \frac{1}{M} \sum_{i=1}^{M} (\Delta v_{radar,car2x})_i \tag{3}$$

In case that the signal $\Delta v_{radar,car2x}$ has a higher rate of change in one time step than the threshold formed by the moving average multiplied with a constant factor c, an abnormal behavior is detected:

$$\Delta v_{radar,car2x} \geq ma \cdot c = th_{IDS} \tag{4}$$

2) Scenario Block Implementation: A lane change by the ACSF shall be suppressed automatically by our developed IDS when it detects a critical situation, before the lane change maneuver begins. We implement the test scenario for the ACSF as a state machine after [18] with input, output, and local variables as shown in Figure 6. The numbers in parentheses after the state names represent integer values to provide the current state via the "currentState" output variable. As the vehicle drives autonomously, acceleration and braking actions by an ideal test-driver within the sub-states are not needed. The guards are implemented as boolean expressions. For example, it is expressed as "RealDistanceToORU $>$ 60.0 && RealDistanceToORU $<$ 62.0" to check whether the start conditions fulfilled in order to execute a transition to the state

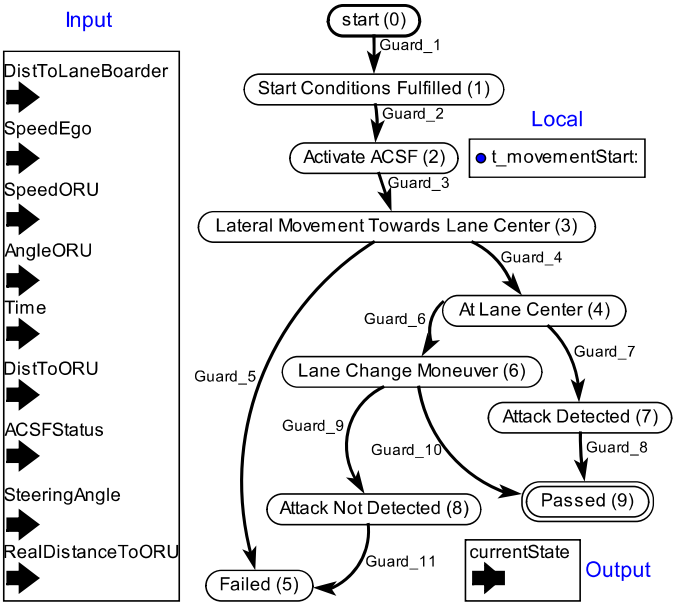

Figure 6. State machine of the environment component "Test" (Figure 5).

"Activate ACSF (2)". Guard_3 triggers the transition to the next state, when the vehicle begins moving to the lane center (determined by the signal *DistToLaneBoarder*). From here it should reach the next state "At Lane Center (4)" within 2 s, otherwise the test fails. If the global variable "attackActive" is set to true (see section 5), the state "Attack Detected (7)" must be reached to pass the test. Otherwise, the test fails. If "attackActive" is set to false, executing a lane change maneuver is right and the test is passed.

C. Results

Figure 7 shows three subplots of the signal courses obtained from our simulation. The time section refers to the period of

the maneuver in which the attack (manipulation of the radar signal at 13.4 s := t_{attack}) takes place.

The first subplot shows the distances of two radar signals: the real one D and the attacked one $D_{attacked}$ (refer to V-B). The manipulated signal is limited and never goes below the critical distance.

In the second subplot, the red signal shows $\Delta v_{radar,car2x}$ (refer to Equation 2) and the green signal th_{IDS} (refer to Equation 4) with the empirically determined factor c = 4. Until t_{attack}, th_{IDS} is above $\Delta v_{radar,car2x}$. Afterward, $\Delta v_{radar,car2x}$ crosses th_{IDS} and lies above it. At this point, an attack must be presumed.

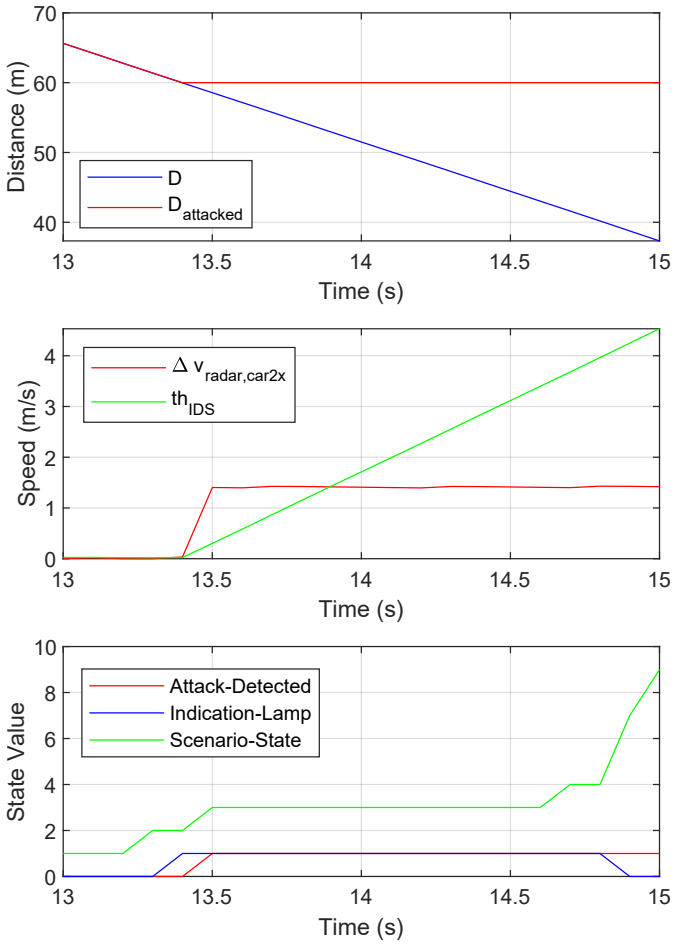

Figure 7. Signal courses of the simulated driving maneuver with successfully detected attack (Cyber-Attack at 13.4 s).

The third subplot illustrates the course of three state signals of the simulation. *Scenario-State* refers to the current state of the scenario state machine, shown in Figure 6 and marked as integer numbers in parentheses after the state names. The signal *Indication-Lamp* indicates whether the ACSF function has been activated (1) in state 2. At 13.5 s, the deviation of the two signals from the third subplot trigger a state transition from 0 to 1 of the signal *Attack-Detected* which indicates that an anomaly is detected. The EV continues heading towards the center line and checks before crossing whether an attack has taken place. It was implemented in this way to detect attacks that occur later, too. At the lane center (*Scenario-State* = 4), a transition to *Scenario-State* = 7 is triggered, as an attack was detected at 13.5 s. As everything was correct, a transition to the final state "Passed (9)" is executed.

In the alternative scenario, we delayed the Car-2-X signal v_{ORU} by 2 s within the block TCU (refer to Figure 5) to simulate a processing latency from the ORU through a server to the EV. The plots are shown in Figure 8. We note that the original signal $\Delta v_{radar,car2x}$ (red curve in the first sub-plot) is never above the threshold th_{IDS} (green curve) and thus the attack is not detected. This is why the signal *Attack-Detected* doesn't change from 0 to 1 and the final state of *Scenario-State* is "Failed (5)", because a lane change is executed and finally leads to a collision between the EV and the ORU.

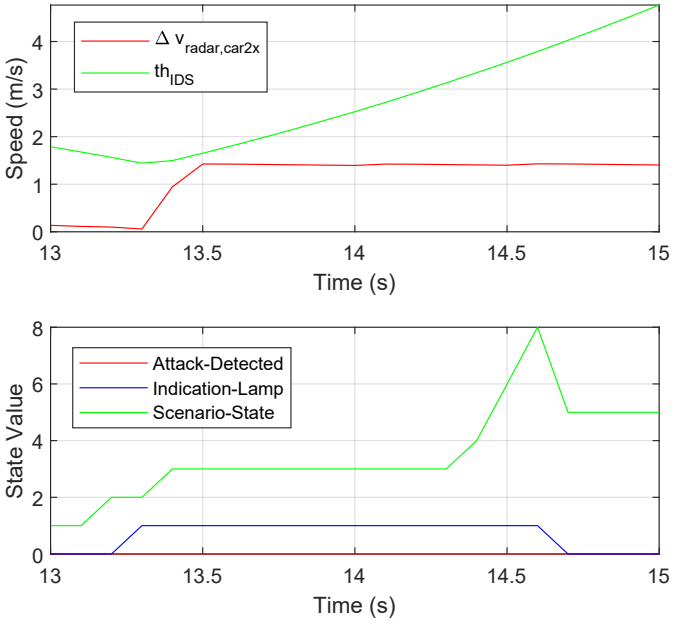

Figure 8. Signal courses of the alternative simulated driving maneuver with failed attack detection and resulting collision (Cyber-Attack at 13.4 s).

D. Discussion

With the two scenarios presented, we were able to show that an attack can be detected with our IDS approach based on Car-2-X data in an ideal scenario derived from UNRs. Moreover, by taking the HA into account, we could selectively modify signals to investigate misuse cases that do not match the ideal case. Based on this, the recognition algorithm can be improved iteratively, for example by adjusting c or by not accepting Car-2-X data with a current timestamp older than 2 s. Instead, the EV may go into a safe state and forbid the lane change due to missing safety information, respectively hand over the control to the driver. In our study, we also found out that the chosen UNR scenario is not sufficient for testing security measures, since, e.g., inconstant speeds of ORU and EV can also affect the detection rate of the IDS.

VI. Conclusion & Future Work

In this paper, we highlight the benefits of using a simulation-based test environment to verify ADAS functions regarding security aspects in an early stage. The framework uses a co-simulation to address three necessary simulation requirements (vehicle models, environment and network communication). One of the main benefits of our approach is the capability to simulate different design variants in order to evaluate software functions or security mechanisms. In this way, boundary conditions such as latencies can be mapped in order to investigate their impact on security. With a use case based on an automated driving function (ACSF) specified by a UNR, we demonstrate the development of a security measure (IDS) by using the simulation platform. This measure allows to detect the specified misuse case in terms of manipulation of the radar signal by using Car-2-X information from the environment in order to avoid the vehicle causing a dangerous driving maneuvers. The corresponding evaluation is accomplished by means of two test maneuvers, where we show how the IDS can be iteratively optimized taking hardware aspects into account. Moreover, we show the need for security-related test-scenarios. We therefore recommend that Threat Analysis and Risk Management (TARA) methods are considered in future UNRs. As a future work, we want to automatize and extend our approach with more details, e.g. several bus-communications. Furthermore, the investigation of the scalability of our approach regarding large-scale E/E architectures and an interweaving with a real test field for the calibration and assessment of the simulations are planned.

Acknowledgment

This work has been developed in the project Profilregion Mobilitätssysteme Karlsruhe, which is partly funded by the state ministries of Baden-Württemberg for Science, Research and Arts and Economy, Labor and Housing.

References

[1] Strategy&, "The 2019 strategy& digital auto report: Time to get real: Opportunities in a transforming market," 2019.

[2] SAE J3016, "Taxonomy and definitions for terms related to driving automation systems for on-road motor vehicles," 2018.

[3] G. Doll, E. Ebel, K. Heineke, M. Kellner, and C. Wiemuth, "Private autonomous vehicles: The other side of the robo-taxi story," 2020.

[4] UNECE, "Regulation no. 79 revision 4," Nov. 2018.

[5] W. Wachenfeld and H. Winner, "Die Freigabe des autonomen Fahrens," in *Autonomes Fahren*, M. Maurer, J. C. Gerdes, B. Lenz, and H. Winner, Eds. Berlin: Springer Vieweg, 2015, pp. 439–464.

[6] Upstream Security Ltd., "Upstream security's 2020 global automotive cybersecurity report," 2020.

[7] O. Y. Al-Jarrah, C. Maple, M. Dianati, D. Oxtoby, and A. Mouzakitis, "Intrusion detection systems for intra-vehicle networks: A review," *IEEE Access*, vol. 7, pp. 21 266–21 289, 2019.

[8] T. Streichert and M. Traub, *Elektrik/Elektronik-Architekturen im Kraftfahrzeug: Modellierung und Bewertung von Echtzeitsystemen*, 1st ed., ser. VDI-Buch. Springer-Verlag, 2012.

[9] H. Czichos, *Mechatronik*. Springer, 2019.

[10] ZHAW, "Von der Mechatronik zu Cyber-physikalischen Systemen," accessed: Mar. 22, 2021. [Online]. Available: https://blog.zhaw.ch/industrie4null/2017/02/06/von-der-mechatronik-zu-cyber-physikalischen-systemen/

[11] S. Harris and F. Maymi, *CISSP All-in-One Exam Guide*. New York: McGraw-Hill Education, 2016.

[12] M. Tepic, "Host-based intrusion detection to enhance cybersecurity of real-time automotive systems," Master's Thesis, Universität Stuttgart, 2019.

[13] Claudius Ptolemaeus, *System Design, Modeling, and Simulation using Ptolemy II*. Ptolemy.org, 2014. [Online]. Available: http://ptolemy.org/books/Systems

[14] German Research Center for Artificial Intelligence, "Open source driving simulation," accessed: Mar. 10, 2021. [Online]. Available: https://opends.dfki.de/

[15] Vector Informatik GmbH, "PREEvision – this is model-based E/E engineering," accessed: Mar. 10, 2021. [Online]. Available: https://www.vector.com/int/en/products/products-a-z/software/preevision/

[16] H. Bucher and J. Becker, "Electric circuit- and wiring harness-aware behavioral simulation of model-based E/E-architectures at system level," in *2018 IEEE International Systems Engineering Symposium (ISSE)*. IEEE, 2018, pp. 1–8.

[17] H. Bucher, C. Reichmann, and J. Becker, "An integrated approach enabling cross-domain simulation of model-based E/E-architectures," in *SAE Technical Paper Series*. SAE International, 2017.

[18] K. Neubauer, H. Bucher, B. Haas, and J. Becker, "Model-based development and simulative verification of logical vehicle functions using executable UN/ECE regulations," in *Proceedings of the 2020 Summer Simulation Conference*, 2020, pp. 1–12.

[19] The Eclipse Foundation, "openPASS platform concept," accessed: Mar. 10, 2021. [Online]. Available: https://openpass.eclipse.org/architecture/#top-level-architecture

[20] S. Gerstl, "Simulation von Fehlersituationen: Fault Injection einfach automatisieren," *Embedded Software Engineering*, accessed: Apr. 09, 2021. [Online]. Available: https://www.embedded-software-engineering.de/simulation-von-fehlersituationen-fault-injection-einfach-automatisieren-a-679627/

[21] E. Aliwa, O. Rana, C. Perera, and P. Burnap, "Cyberattacks and countermeasures for in-vehicle networks," *ACM Computing Surveys*, vol. 54, no. 1, pp. 1–37, 2021.

[22] F. Sommer, J. Dürrwang, and R. Kriesten, "Survey and classification of automotive security attacks," *Information*, vol. 10, no. 4, p. 148, 2019.

[23] W. Wu, Y. Huang, R. Kurachi, G. Zeng, G. Xie, R. Li, and K. Li, "Sliding window optimized information entropy analysis method for intrusion detection on in-vehicle networks," *IEEE Access*, vol. 6, pp. 45 233–45 245, 2018.

[24] S.-F. Lokman, A. T. Othman, and M.-H. Abu-Bakar, "Intrusion detection system for automotive Controller Area Network (CAN) bus system: a review," *EURASIP Journal on Wireless Communications and Networking*, vol. 2019, no. 1, 2019.

[25] C. Young, J. Zambreno, H. Olufowobi, and G. Bloom, "Survey of automotive controller area network intrusion detection systems," *IEEE Design & Test*, vol. 36, no. 6, pp. 48–55, 2019.

[26] G. Loukas, T. Vuong, R. Heartfield, G. Sakellari, Y. Yoon, and D. Gan, "Cloud-based cyber-physical intrusion detection for vehicles using deep learning," *IEEE Access*, vol. 6, pp. 3491–3508, 2018.

[27] M. Weber, S. Klug, E. Sax, and B. Zimmer, "Embedded hybrid anomaly detection for automotive CAN communication," in *9th European Congress on Embedded Real Time Software and Systems (ERTS 2018)*, 2018.

[28] D. K. Oka, "Fuzz testing virtual ECUs as part of the continuous security testing process," *SAE International Journal of Transportation Cybersecurity and Privacy*, vol. 2, no. 2, 2019.

[29] M. Felderer, P. Zech, R. Breu, M. Büchler, and A. Pretschner, "Model-based security testing: a taxonomy and systematic classification," *Software Testing, Verification and Reliability*, vol. 26, no. 2, pp. 119–148, 2016.

[30] M. Appel, P. S. Oruganti, Q. Ahmed, J. Wilkerson, and R. Sekar, "A safety and security testbed for assured autonomy in vehicles," in *SAE Technical Paper Series*. SAE International, 2020.

[31] S. Raue, "Systemorientierung in der modellbasierten modularen E/E-Architekturentwicklung." Universität Tübingen, 2019.

[32] C. Miller and C. Valasek, "Remote exploitation of an unaltered passenger vehicle," *Black Hat USA*, vol. 2015, 2015.

[33] OICA and CLEPA, "ACSF test procedure draft proposal – for discussion," 2015.