2022 30th Euromicro International Conference on Parallel, Distributed and Network-Based Processing (PDP) | 978-1-6654-6958-6/22/$31.00 ©2022 IEEE | DOI: 10.1109/PDP55904.2022.00047

SECPAT: Security Patterns for Resilient Automotive E/E Architectures

Christian Plappert, Florian Fenzl, Roland Rieke
Fraunhofer SIT, Germany
{firstname.lastname}@sit.fraunhofer.de

Ilaria Matteucci, Gianpiero Costantino, Marco De Vincenzi
CNR, Italy
{firstname.lastname}@iit.cnr.it

***Abstract*—Automated driving requires increasing networking of vehicles, which in turn broadens their attack surface. In this paper, we describe several security design patterns that target critical steps in automotive attack chains and mitigate their consequences. These patterns enable the detection of anomalies in the firmware when booting, detect anomalies in the communication in the vehicle, prevent unauthorized control units from successfully transmitting messages, offer a way of transmitting security-related events within a vehicle network and reporting them to units external to the vehicle, and ensure that communication in the vehicle is secure. Using the example of a future high-level Electrical/Electronic (E/E) architecture, we also describe how these security design patterns can be used to become aware of the current attack situation and how to react to it.**

***Index Terms*—automotive threat mitigation and resilience, automotive security, cybersecurity engineering, intrusion detection, connected car, AUTOSAR, Trusted Platform Module (TPM), Device Identifier Composition Engine (DICE)**

I. Introduction

Automated driving makes traffic safer, but requires vehicles to be increasingly connected. However, this networking expands the possibilities for attacks on a vehicle. Trend Micro Research [1] analyzed four prominent attacks, namely the Jeep hack of 2015, the TESLA hacks of 2016 and 2017, and the BMW hack of 2018. The generic chain of attacks from this analysis shows that these attacks initially attack the vehicle's head unit via WiFi or the mobile network, then exploit a weak point there, for example in the webkit, and then carry out a root exploit on the operating system. The Controller Area Network (CAN) gateway is then attacked and new firmware is uploaded there, whereupon the gateway is reprogrammed in order to finally take over other Electronic Control Units (ECUs). Fake messages can be sent over the CAN bus now or in a previous step. Some steps in this chain have also been demonstrated by a hack on the headunit of a KIA Cee'd in 2020 [2] and the TBONE Tesla hack with entry over WiFi in 2021 [3].

For new Electrical/Electronic (E/E) vehicle architectures, we assume that there is at least one additional entry point for attackers through the communication during electrical charging and since the internal vehicle communication is expanded to include Automotive Ethernet, there will also be further possible steps in the vehicle-internal attack chain.

In this paper, we describe several security design patterns with the aim of targeting critical steps in such attack chains and mitigating their consequences. We also describe, using an example of a future high-level E/E vehicle architecture, how these security design patterns can be used to become aware of the current attack situation and to react to it.

The paper is structured as follows. We describe necessary background and related work in Section II. In Section III we introduce our approach and detail about our patterns consisting of a host-based integrity verification pattern (Section III-A), an Intrusion Detection System (IDS) pattern (Section III-B), a challenge-based intrusion prevention pattern (Section III-C), an alert reporting pattern (Section III-D), and a pattern for securing the in-vehicle communication (Section III-E). In Section IV we map the patterns capabilities to typical design principles of systems security engineering and in Section V we show how the proposed patterns can be applied to a reference automotive E/E architecture. Section VI concludes our paper.

II. Background and Related Work

The related work for this paper is manifold. Existing work on generic security design patterns has been adopted for many application domains. However, work on this approach for specific pattern to be applied in automotive systems is rare. In [4], Martin et al. propose a pattern-based approach that interlinks safety and security patterns. The application is demonstrated by an automotive case study but describes only very few examples. In [5], Cheng et al. describe a collection of security design patterns targeted to the automotive domain. They leverage an earlier security pattern template from [6] that is tailored to secure systems development. Cheng et al. extend this template to include fields specific to the automotive domain and SAE J3061 [7] cybersecurity guidelines which is now superseded by ISO/SAE FDIS 21434 [8]. They exemplified the usability of their approach by a collection of ten security design patterns for the automotive domain.

In our work, we leverage the pattern template from Cheng et al. to specify several security patterns for use in new E/E vehicle architectures. In particular, these patterns address the areas of in-vehicle intrusion detection and prevention as well as intrusion reporting and deployment of models. With respect to these areas we take into account recent work on standardization as well as best practice, such as the ENISA report on good practices for security of smart cars [9]. We also aim at compatibility with the Automotive Open Systems

978-1-6654-6958-6/22/$31.00 ©2022 IEEE
DOI 10.1109/PDP55904.2022.00047

Architecture (AUTOSAR) standard. AUTOSAR specifies an in-vehicle IDS protocol [10] for the transmission of qualified security events from an IDS manager instance to an IDS reporter instance, and an IDS manager for adaptive platform [11] as well as a specification of Secure Onboard Communication (SecOC) [12].

III. Design Patterns

As described above, for the description of our patterns we leverage the automotive security pattern template defined in [5] with the following elements: *a) Pattern Name*, *b) Intent* describing the underlying security problem addressed by the pattern, *c) Motivation*, *d) Properties* in terms of STRIDE, *e) Applicability*, *f) Structure* given by a Unified Modeling Language (UML) class diagram, *g) Behavior* defined by a UML sequence diagram, *h) Constraints*, *i) Consequences* expressed by security properties, performance, cost, manageability, usability, *j) Known Uses*, and *k) Related Patterns*.

In the following, we describe the new security patterns for use in resilient attack detection and mitigation concepts for future E/E architectures.

Host-based Integrity Verification System (HIVS): This pattern provides a mechanism to detect anomalies (unauthorized modifications) in ECU software and firmware at boot time. The pattern has two variants using the Trusted Platform Module (TPM) as hardware trust anchor or the Device Identifier Composition Engine (DICE) as lightweight alternative.

Network-based Intrusion Detection System (NIDS): This pattern provides a mechanism to detect anomalies in (automotive) network communication by using metadata and extracted payload features from observed ECU messages.

Challenge-based Intrusion Prevention System (CIPS): This pattern provides a mechanism to prevent anomalies in the (automotive) network communication by using a challenge based on the knowledge of the payload to authenticate an ECU.

Security Event Reporting (SER): This pattern provides a way to aggregate security relevant events, such as software anomalies or unexpected network traffic, within a vehicle network and report it to an outside Security Operation Center (SOC).

Secure CAN Communication (SCC): This pattern provides a mechanism to secure the in-vehicle communication. It guarantees confidentiality, integrity, and authentication.

A. Host-based Integrity Verification System (HIVS)

Intent: This pattern provides a mechanism to detect anomalies (unauthorized modifications) in ECU software and firmware at boot time. Starting from an initial Root of Trust for Measurement (RoTM) (e.g., in the bootloader), the later going-to-be-executed parts of the software are measured (by calculating a hash value of the software component) before handing control to them. This creates a measurement chain covering the whole software state during boot time that could then be reported to a remote entities like a SOC, an Original Equipment Manufacturer (OEM) backend or other Vehicle to Everything (V2X) participants. In this pattern, two instantiations with different security guarantees for a HIVS are proposed that respect the heterogeneous capabilities of ECUs within the vehicle: a TPM-based [13] approach where the TPM as additional hardware security chip secures the integrity verification and a software-based solution utilizing DICE [14] as a lightweight RoTM. Both options are applicable to the automotive domain since TPMs are already utilized in vehicles [15] and DICE is easy and cost-effective to integrate since it has minimal silicon requirements. The differences of both options are explained throughout the section.

Motivation: Vehicle networks are complex and their compromise of high revenue to attackers. However, trusted relationships within the vehicle and to its environment are crucial, e.g., trust in authentic messages may be safety-critical.

Properties: The HIVS pattern can be used to satisfy the integrity properties of ECUs for remote verifiers.

Applicability: The HIVS pattern is applicable to attack detection and can be enhanced to be applicable to attack prevention and mitigation, e.g., by sealing decryption or authentication keys to the software state.

Structure: The pattern structure is depicted in Fig. 1 for the TPM-based and in Fig. 3 for the DICE-based instantiation of the HIVS pattern. Both pattern instantiations consist roughly of two main components: the host sensor and the Reporting Manager (RM). The RM is part of the SER pattern (cf., Section III-D) and thus links the two patterns. In both cases, the host-based sensor is assigned to a specific ECU. For the TPM-based HIVS pattern, the host sensor is subdivided into: Root of Trust for Measurement (RoTM), Root of Trust for Storage (RoTS), Root of Trust for Reporting (RoTR), log database, and the Software Component (SWC). RoTS and RoTR are implemented by the TPM. The DICE-based alternative of the host-based sensor is simpler and only consists of RoTM and SWC.

Behavior: In both HIVS pattern instantiations, the initial RoTM, typically the CPU, is responsible for the initial measurement of the first SWC (SWC1). After the measurement is done, it hands control to SWC1 that will continue the measurement process with the next component (SWC2). This will create a measurement chain that covers the whole boot process and represents the system state at boot time. The system state can then be used to seal data or can be reported to a remote verifier that than can verify the trustworthiness of the system (remote attestation).

In the TPM-based instantiation depicted in Fig. 2, the single measurement will be extended as binary hash values into the RoTS and, enriched with some meta data, in the log database. In case of remote attestation, the measurement values are authenticated by the RoTR using authentication mechanisms like signatures or Message Authentication Codes (MACs) before leaving the RoTS. The DICE-based instantiation depicted in Fig. 4 is simpler and also its concept is a bit different. Instead of storing integrity measurements of the SWCs throughout the boot sequence in a RoTS, the measurements are used as inputs for key derivation functions to create the so-called Compound Device Identifier (CDI) keys. A Unique Device Secret (UDS)

serves as initial input to derive device specific keys. Thus, the integrity of the system can be verified if the correct keys were derived, e.g., by the RM starting a challenge-response scheme.

Either actively or upon request by a remote verifier, the authentic events (answered challenges or measurements with event logs) are collected by the RM and send back via the Reporter. They can then be used to verify the software state and decide if the component is trustworthy.

Constraints: For both pattern instantiations, cryptographic algorithms during boot time are quite lightweight (hash operations) in terms of storage and performance. However, for the TPM-based instantiation, the creation of log data base may be quite heavy depending on the complexity and granularity of the measured component. However, this could be outsourced to the remote party that preventively stores the entries. During reporting, the measurements are authenticated in both variants. Depending on the reporting frequency and cryptographic algorihm, this may have some performance overhead. A solution to make the reporting for the TPM instantiation more lightweight is presented in [16].

Consequences: Table I describes the consequences.

TABLE I
CONSEQUENCES FOR THE HIVS PATTERN.

Accountability	Accountability is improved as a system can trust in a measurement chain covering the software state during boot time.
Confidentiality	Not directly addressed.
Integrity	Integrity of the system is improved since also remote parties can verify the integrity of the vehicle.
Availability	Not directly addressed.
Performance	Lightweight symmetric cryptographic operations (hashes) during boot time, more resource-intensive (a-) symmetric cryptography (signature/mac generation) during runtime.
Cost	Additional hardware cost for the TPM instantiation.
Manageability	Not directly addressed.
Usability	Some overhead regarding performance and cost (TPM chip) may be introduced to the system.

Known Uses: Both Trusted Computing Group (TCG) and Internet Engineering Task Force (IETF) standardize attestation schemes for both HIVS variants, e.g., in [17], [18], while benefiting automotive security solutions are for example described in [19] for secure updates and in [20] for feature activation.

Related Patterns: The *Security Event Reporting* pattern is directly related and can be used to collect the reports and send them to remote parties. The *Tamper Resistance* and *Third-party Validation* patterns in [5] partially address similar security goals.

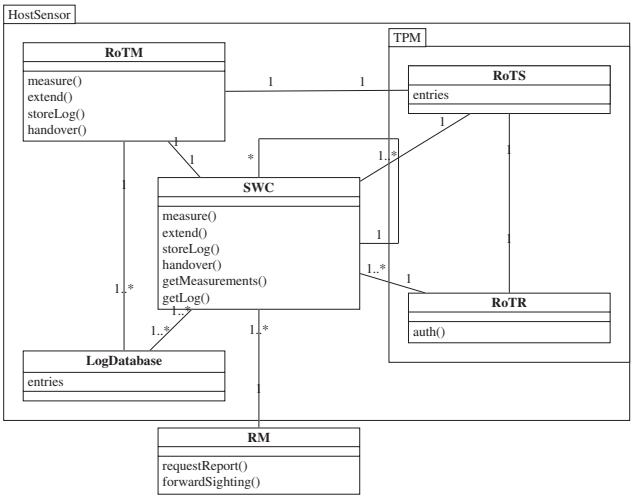

Fig. 1. Structure of the HIVS pattern with TPM

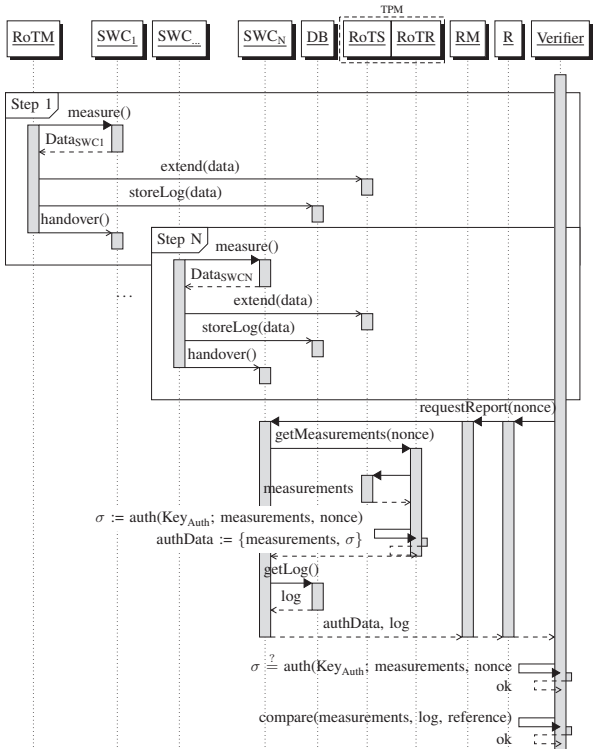

Fig. 2. Behavior of the HIVS pattern with TPM

B. Network-based Intrusion Detection System (NIDS)

Intent: This pattern provides a mechanism to detect anomalies in (automotive) network communication by using metadata and extracted payload features from ECU messages.

Motivation: Communication within a modern vehicle is more and more connected to the outside world, which provides attackers with a multitude of new approaches to interfere and manipulate the internal communication within the vehicle network. To detect known intrusion signatures, as well as unknown anomalies in data traffic, a dedicated system is required that can classify and verify the integrity of messages and adapt to new situations.

Properties: The NIDS can influence the integrity as well as the authentication property for observed components.

Applicability: The NIDS is applicable to attack detection. When integrated with an appropriate system, it can also be

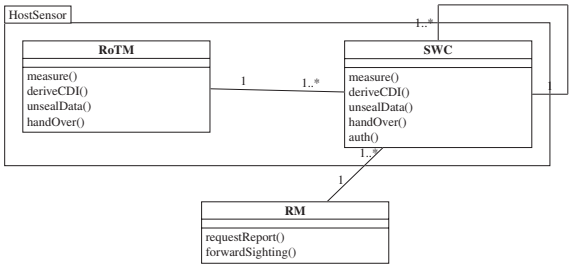

Fig. 3. Structure of the HIVS pattern with DICE

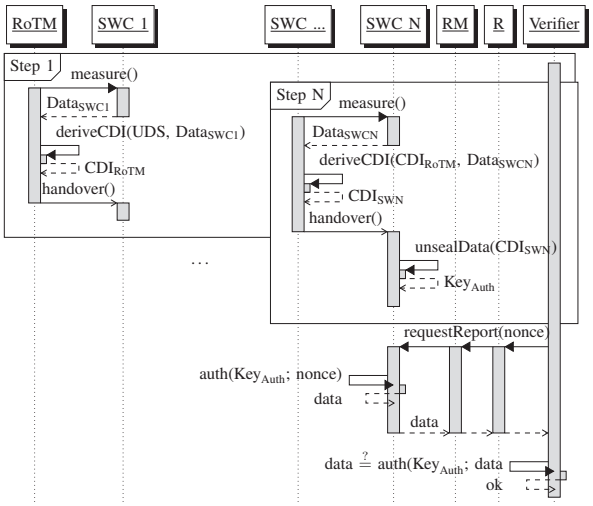

Fig. 4. Behavior of the HIVS pattern with DICE

used for attack prevention and mitigation.

Structure: A Network Sensor, as depicted in Fig. 5, is assigned to a specific subnet or component on a network. The Classifier Database provides a set of Classifier models used in classification of anomalies on the relevant system. The Classifier may be any type of either machine learning model [21], [22] or rule-based model [23] depending on the specific requirements of the current subsystem, such as the complexity of the expected data or the computation capabilities [24]. Classifiers are selected by the Network Sensor and may be used in any combination with one another with the sensor being able to determine the necessary models for the current situation. The Network Sensor can be set to listen for all available network traffic and forward detection events. The RM accumulates detection events from an arbitrary number of assigned Network Sensors and forwards sightings to a central reporting unit within the same network or an external SOC.

Behavior: As shown in Fig. 6, the Network Sensor is observing the network for new messages. As a new message is received, the respective classifiers are retrieved from the Classifier Database and used to classify the message. Depending on the classifier result an Event is raised to the RM.

Constraints: In a real time environment, such as an automotive network, the classification of messages is required

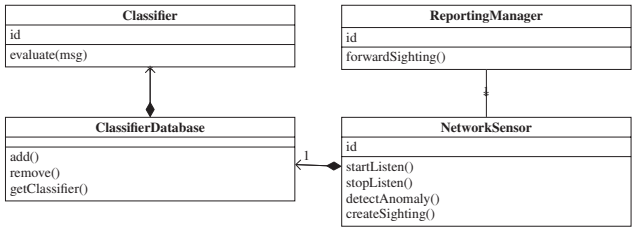

Fig. 5. Structure of the NIDS pattern

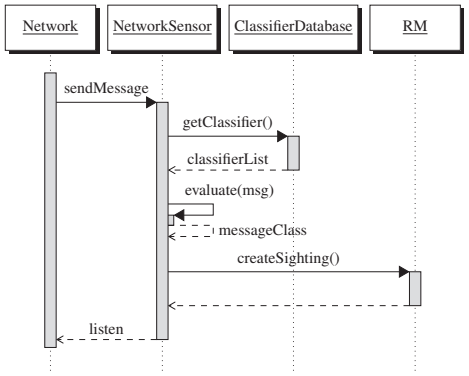

Fig. 6. Behavior of the NIDS pattern.

to be fast and efficient. Complex machine learning evaluations of every observed package are often not possible to perform within real time constrains. This is especially the case using neural network models without dedicated hardware acceleration components, such as a Tensor Processing Unit (TPU), that may increase the cost of the component significantly. The available resources are therefore a limiting factor to the complexity of the classification process.

Consequences: Table II describes the consequences.

TABLE II
CONSEQUENCES FOR THE NIDS PATTERN.

Accountability	Depends on type of IDS.
Confidentiality	Not addressed.
Integrity	Can be improved if ECU impersonation is detected.
Availability	Might be improved if e.g., DoS messages can be deleted early but is reduced by overhead of classification.
Performance	Classification cost is depending on specific algorithms and hardware.
Cost	Additional hardware to classify traffic may incur a cost.
Manageability	RM controls sensors and accumulates all intrusion information.
Usability	General safety requirements such as hard real time requirements and encryption might influence usability of the pattern.

Known Uses: This pattern can be used in a similar way as the AUTOSAR specifications in [10], [11].

Related Patterns: The *Security Event Reporting* pattern is directly related and can be directly integrated. This pattern is also related to the Signature-based IDS pattern proposed by Cheng et al. [5] and can be integrated alongside to increase the detection rate of anomalies.

C. Challenge-based Intrusion Prevention System (CIPS)

Intent: This pattern provides a mechanism to prevent unauthorized ECUs to deliver messages by using knowledge challenges on how to generate valid messages.

Motivation: Modern vehicles are computers on wheels connected to the Internet. Thus, they may be vulnerable to local and remote cyberattacks aiming at altering information and communication among the ECUs. This pattern aims at preventing that an attacker sends malicious CAN frames among different partitions of the intravehicle network.

Properties: CIPS can be used to satisfy both integrity of messages and authentication of the sender.

Applicability: CIPS is applicable to attack prevention and mitigation. It is able to detect attacks, such as *fuzzing* and *replay* attacks. It may also be used for attack detection.

Structure: The structure is depicted in Fig. 7. The involved participants are the ECUs of the intravehicle network. We consider a sender ECU, aka the Resolver, and a partition Gateway (GW), i.e., the Challenger, that enforces the prevention mechanism. To successfully send a frame via Challenger to the destination partition, the Resolver must first successfully answer a challenge of the Challenger.

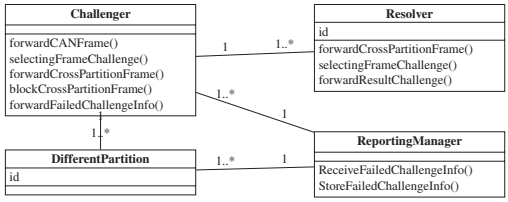

Fig. 7. Structure of *Challenge-based Intrusion Prevention System (CIPS)* pattern.

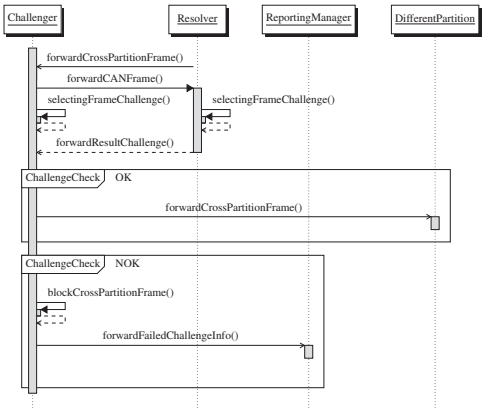

Fig. 8. Behavior *Challenge-based Intrusion Prevention System (CIPS)* pattern.

Behavior: In the CIPS pattern, the Challenger intercepts all *cross-partition* CAN frames that are generated from Resolvers in his partition, e.g., untrusted partition, and addressed to one or more ECUs in another partition, e.g., trusted partition. The pattern aims to mitigate attacks by verifying the authenticity of the Resolver by means of a preceding challenge method. Thus, each Resolver has to know 1) the *Database of CAN messages* (DBC), which is proprietary to a particular OEM and can be considered as a long-term secret only known by the Challenger and legitimate Resolvers, 2) the *challenge set* that represents the type of challenges that the Challenger will ask the Resolver to authenticate, and 3) the *encoding generation method* that defines how frames, needed for the challenge, must be generated by the Resolver that sent a cross-partition frame.

The Challenger challenges the Resolver asking for a challenge, sending a CAN frame, as part of the handshake protocol (Fig. 8). In case the Resolver correctly answers the challenge, i.e., by sending the correct frame of the DBC asked by the Challenger, the Challenger has authenticated the Resolver ECU. In fact, only legitimate Resolvers know the full DBC, the set of challenges, and the encoding method. Hence, only legitimate Resolvers are able to provide the correct answer. In this case, the message of the Resolver is authorized to be forwarded to the original cross-partition frame. In case of an incorrect answer, the frame is discarded. Note that, a Resolver with a partial knowledge of the DBC is not able to properly answer to the challenge since the pattern requires full knowledge of the DBC. The answer of the challenge is also sent to the RM to take trace of the possible attacks.

Constraints: In a real in-vehicle network, challenging the ECU has to be made in a fast and efficient way.

Consequences: see Table III.

TABLE III
CONSEQUENCES OF THE CIPS PATTERN.

Accountability	Depends on type of CIPS.
Confidentiality	Not addressed.
Integrity	Can be improved if ECU impersonation is detected.
Availability	Might be improved if one or more challenges fail in case of an attack, e.g., DoS attack. Once identified the involved frames are not forwarded to the destination partition.
Performance	CIPS can be run on low-power ECUs, however, more powerful ECUs can improve the challenge performance.
Cost	Additional hardware with improved performances may incur a cost.
Manageability	RM controls frames sent by ECUs and stores all failed challenge information.
Usability	General safety requirements, such as hard real time requirements, might influence the usability of the pattern.

Known Uses: A possible implementation and application of CIPS is given in [25].

Related Patterns: Both *Reporting Manager* and *NIDS* patterns are related to CIPS. Moreover, CIPS is related to the *Firewall* and the *Multi-Factor Authentication* pattern in [5].

D. Security Event Reporting (SER)

Intent: This pattern provides a way to aggregate security relevant events, such as software anomalies or unexpected network traffic, within the vehicle network and to report it to an outside SOC.

Motivation: A solely autonomous anomaly detection software is in most cases not feasible within an automotive network. This is especially the case when mitigation operations are set to occur as a response to critical security events. An external SOC is required to oversee mitigation actions and adapt classifiers to new situations. For this, security events from multiple vehicles have to be accumulated and verified by human operators.

Properties: The Security Event Reporting (SER) pattern can influence the integrity and non-repudiation properties.

Applicability: The SER pattern is applicable to detection, prevention and mitigation.

Structure: The structure is depicted in Fig. 9. The Reporter component should be located on the primary component responsible for outside connection, e.g., the Telematic Control Unit (TCU). It would also be possible to update the detection models and rules of each intrusion sensor through this component. Security components are systems capable of reporting security events, such as observed anomalies, in a structured form called Security Event. A Security Event is a uniquely identifiable report on a singular security relevant observation with a meaningful description and an exact time of occurrence. These Security Events can then be accumulated into uniquely identifiable Security Reports, which can then be used to forward information compressed to Remote Receiver, e.g., SOC or OEM backend.

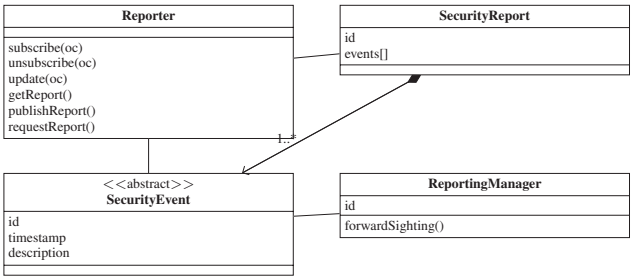

Fig. 9. Structure of the SER pattern.

Behavior: The specific behavior is described by Fig. 10. Whenever the Reporter component receives a sighting from any Reporting Manager within the network, either as a direct response or as a timed event, a report of accumulated sightings is created. When such a report is created, all known SOCs are notified and may then request the sighting report. It is also possible for every SOC to actively request the latest report from a vehicle Reporter component.

Constraints: The Reporter component provides intruders with a potential centralized point of attack. For the Security Events send from a RM to the Reporter additional verification steps may be required to ensure integrity.

Consequences: See Table IV.

Known Uses: In particular, Metzker et al. presented their ideas, concepts and software architecture proposals concerning IDS and intrusion reporting in [26], while AUTOSAR provides a specification of an in-vehicle IDS protocol in [10] with a use case which covers event reporting.

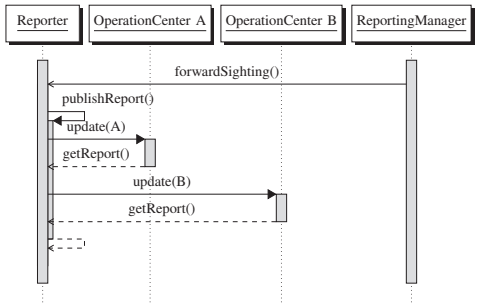

Fig. 10. Behavior of the SER pattern.

TABLE IV
CONSEQUENCES OF THE SER PATTERN.

Accountability	The reporting component allows for the system to be accountable for the non-repudiation of traffic events inside the vehicle.
Confidentiality	Security event reporting shares vehicle internal traffic including potential personal data with external entities, which may prevent confidentiality without appropriate data security measures.
Integrity	By reporting detected anomalies, the security event reporting may enable appropriate mitigation measures to be taken.
Availability	The security event reporting may prevent availability in the vehicle network depending on the number of security events raised.
Performance	Depending on the urgency of detected anomaly sightings there is an additional load on the vehicle network, while the reporter itself requires low computational power.
Cost	The system itself requires low computational performance and can be integrated into an existing system with required outside connections and connection to reporting components.
Manageability	Allows for possible mitigation actions to be more manageable and better directed.
Usability	Not addressed.

Related Patterns: This pattern relates to all intrusion detection patterns, such as the HIVS, NIDS, and CIPS pattern by providing a way to accumulate the data of all sensors. The Signature-based IDS pattern proposed by Cheng et al. [5] can also be integrated similarly with minimal modification.

E. Secure CAN Communication (SCC)

Intent: Communication among the in-vehicle ECUs enable the functionalities of a vehicle. The goal of this pattern is to guarantee communications are secure.

Motivation: In-vehicle protocols may lack of security mechanisms. The most famous one is the CAN bus protocol that allows an ECU to broadcast messages in clear among all ECUs in (a partition of) the in-vehicle network. This lack of security mechanisms may lead to several well-known attacks such as man in the middle, replay, or sniffing attacks.

Properties: Several mechanisms can be put on top of in-vehicle protocols with the aim to guarantee authentication, data integrity, and confidentiality.

Applicability: The SCC pattern enables ECUs to generate secure communication frames to prevent that they may be re-used, sniffed, generated without proper authorization.

Structure: The structure of the SCC pattern is depicted in Fig. 11. All ECUs belonging to the in-vehicle network are able to generate a secure frame as well as create *vanilla* protocol frames to guarantee backward compatibility. In particular, the participants are the sender ECU, the receiver ECU, the Freshness Value Manager (FVM) that manages the freshness values, and the Crypto Service Manager (CSM) that manages the encryption and decryption keys. The SCC pattern aims to produce secured frame by using 1) a MAC for authentication, 2) the FV for integrity, and 3) encryption for confidentiality.

The security algorithms adopted in the SCC pattern are chosen to not overload the communications timing and performances. In particular, this pattern is designed to meet real-time constraints and do not negatively impact on the safety aspects. Note that, the SCC security properties may be downgraded, i.e., removing the confidentiality mechanism, to obtain more performance. This choice, however, will impact on the pattern security properties.

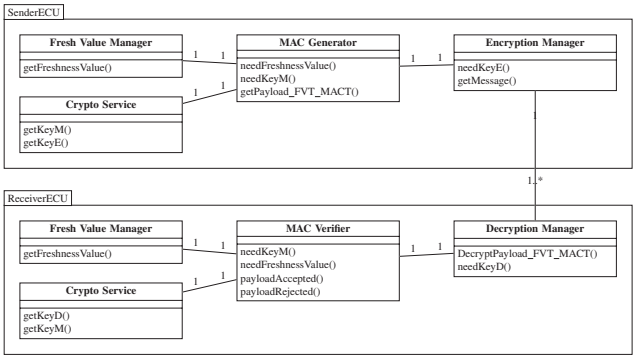

Fig. 11. Structure of the SCC pattern between two ECUs.

Behavior: The behavior of the SCC pattern is depicted in Fig. 12. Before sending a payload, the sender generates the MAC starting from the payload and possibly the *Freshness Value* (FV) calculated according to the Monotonic Counter provided by the Freshness Manager as designed in Fig. 12 (an ECU may decide to ignore the FV). So, the secured CAN frame is composed by the payload, the truncated MACT and, optionally, the truncated freshness value (FVT). Then it is encrypted and sent.

Before accepting the CAN frame, the receiver has to decrypt the CAN frame and validate its authenticity by verifying the MAC. The receiver generates a freshness value for verification (FVV) starting from the Monotonic Counter received by the FVM. Then, it calculates the MAC by using the received payload and the FVV. If the outcome equals the received MAC, then the payload is accepted, otherwise it is discarded.

Despite the fact that generally the Encrypt-then-MAC scheme is preferred, this specification proposes the MAC-then-encrypt approach that has the following benefits in the CAN protocol: 1) the risk of message rebuilding is zeroed

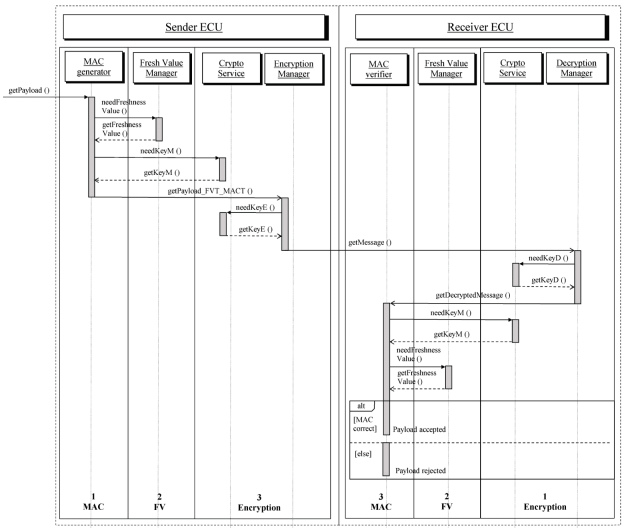

Fig. 12. Behavior of the SCC pattern between two ECUs.

because there is no padding effect due to the fixed length of the considered messages and the used encryption algorithms with 64-bit block size, 2) the transmission of an additional frame to contain the MAC is not needed in case of frames where the 64 bits are already taken.

Constraints: The SCC pattern works by reducing the dimension of the payload of the frame in order to introduce security information such as the MAC and the freshness value. This may lead to the necessity of introducing extra-traffic in the network in case of extremely complex frames.

Consequences: See Table V.

TABLE V
CONSEQUENCES OF THE SCC PATTERN.

Accountability	SCC is accountable for integrity, confidentiality and senders authentication in its maximum instantiation.
Confidentiality	CAN frames exchanged among the ECUs will be encrypted providing data confidentiality on the transmitted payload in case the encryption algorithm is applied.
Integrity	CAN frames exchanged among the ECUs will use an integrity mechanism to identify payload manipulation in case the FV is introduced into the secured frame.
Availability	SCC does not prevent availability even if it may add some performance cost.
Performance	This will depend on the algorithm used in the security pattern and on the hardware of the ECU.
Cost	The security pattern itself works with low computational resource but better results can be obtained with more powerful hardware.
Manageability	Not directly addressed.
Usability	Low overhead regarding performance may be introduced to the system.

Known Uses: The work in [25] describes a possible design and deployment of a basic software module called CINNAMON. It has been designed according to the guidelines depicted by the AUTOSAR classic platform in such a way that its integration will result quite straightforward. An example of a lightweight version of the SCC pattern is the AUTOSAR

TABLE VI
MAPPING THE PATTERN CAPABILITIES TO THE NIST DESIGN PRINCIPLES [27]

No	Title	Description	HIVS	NIDS	CIPS	SER	SCC
		1. Security Architecture and Design					
1.01	Clear Abstractions	Interfaces and functions should be simple and well-defined.	✓	✓	✓	✓	✓
1.02	Least Common Mechanism	Subsystems should refrain from using the same mechanisms for resource access.	-	-	-	✓	✓
1.03	Modularity and Layering	System should be designed modular and layered to reduce complexity.	✓	✓	✓	✓	✓
1.04	Partially Ordered Dependencies	System dependencies should be partially ordered by their dependencies.	✓	✓	✓	✓	✓
1.05	Efficiently Mediated Access	The shared access to resources should be secured.	✓	-	-	-	✓
1.06	Minimized Sharing	Resource sharing between components should be avoided.	✓	-	-	-	✓
1.07	Reduced Complexity	System design should be as simple and small as possible.	✓	✓	✓	✓	✓
1.08	Secure Evolvability	Security guarantees should adapt as system evolves.	✓	✓	✓	✓	✓
1.09	Trusted Components	System design should enable to establish trusted components.	✓	-	✓	-	✓
1.10	Hierarchical Trust	The overall system trust should be derivable from the trust of its components.	✓	-	✓	-	✓
1.11	Inverse Modification Threshold	The degree of component protection relates to its trustworthiness.	✓	-	✓	-	✓
1.12	Hierarchical Protection	A component does not need to be protected from more trustworthy components.	✓	-	-	-	✓
1.13	Minimized Security Elements	Trusted components should be kept as low as possible.	✓	✓	-	✓	✓
1.14	Least Privilege	Components should not have more privileges than necessary for its functionality.	✓	✓	✓	✓	✓
1.15	Predicate Permission	Multiple authorized entities should consent to access or operate on sensitive data.	-	-	-	-	-
1.16	Self-Reliant Trustworthiness	Systems should minimize the reliance on third parties to state their trustworthiness.	-	✓	✓	-	✓
1.17	Secure Distributed Composition	The security of distributed systems should be equal to individual components.	-	✓	-	✓	✓
1.18	Trusted Communication Channels	Communication channels should be secured against unauthorized access.	✓	-	-	-	✓
		2. Security Capability and Intrinsic Behaviors					
2.01	Continuous Protection	All components enforcing the security policy should have uninterrupted protection.	✓	✓	✓	✓	✓
2.02	Secure Metadata Management	The system should protect the metadata necessary for its security guarantees.	✓	-	-	-	-
2.03	Self-Analysis	Components should be able to access their internal state and functionality.	✓	✓	-	✓	-
2.04	Accountability and Traceability	System should be able to trace security-relevant actions.	-	✓	✓	✓	-
2.05	Secure Defaults	The default security policy configuration should be restrictive and conservative.	✓	✓	✓	✓	✓
2.06	Secure Failure and Recovery	A failure or corresponding recovery mechanism should not violate the security.	✓	✓	✓	✓	✓
2.07	Economic Security	A security mechanism should not be costlier than the potential damage.	✓	✓	✓	✓	✓
2.08	Performance Security	A security mechanism should not degrade the system performance unnecessarily.	✓[1]	✓[1]	✓	✓	✓
2.09	Human Factored Security	The user interface should be intuitive and user friendly.	-	-	-	✓	-
2.10	Acceptable Security	System performance & privacy should be compliant with the user expectation.	-	-	-	✓	✓
		3. Life Cycle Security					
3.01	Repeatable & Documented Procedures	It should be ensured that a component can be reconstructed at a later time.	-	✓	✓	✓	✓
3.02	Procedural Rigor	Life cycle security concepts should be appropriate to intended trustworthiness.	-	✓	✓	✓	✓
3.03	Secure System Modification	It should be ensured that system modifications do not decrease security guarantees.	✓	-	-	-	-
3.04	Sufficient Documentation	Documentation for personnel to contribute and not detract the system security.	-	✓	-	✓	-
		4. Approaches to Trustworthy Secure System Development					
4.01	Reference Monitor Concept	System should be at least tamper-proof, inevitable, and verifiable.	✓	✓	✓	✓	✓
4.02	Defense in Depth	Security-sensitive components should be secured by a series of security barriers.	✓[2]	✓[2]	✓[2]	✓[2]	✓[2]
4.03	Isolation	Security-sensitive components should be isolated from the rest of the system.	✓	✓[3]	-	-	✓[3]

[1]: Depends on the specific pattern implementation (e.g., DICE vs. TPM for HIVS or Rule-based vs. Neural Networks for NIDS)
[2]: Defense in depth is achieved by combining the patterns.
[3]: Depends on implementation and integration into system.

SecOC software module [12]. In fact, SecOC does not encrypt the frame. Hence, confidentiality is not guaranteed.

Related Patterns: None of the previous security patterns is related to SCC. Referring to [5], SCC can be considered an extension of the *Symmetric Encryption* pattern.

IV. DESIGN PRINCIPLES

We now show how these pattern relate to design principles for systems security engineering. The ISO/SAE 21434 standard for road vehicles cyber security engineering [8] recommends in its recommendation RC-10-06 that established and trusted design and implementation principles should be applied to avoid or minimize the introduction of weaknesses and it further refers to the design principles for architectural design for cybersecurity in NIST Special Publication 800-160 Vol. 1 [27]. In Table VI we map our security pattern to these generic security design principles which provide the foundation for engineering trustworthy secure systems. Following the NIST taxonomy, we have structured the table into (1) structural design principles that affect the fundamental architecture of the system and interfaces, (2) security capability and intrinsic behaviors subsuming generic patterns for implementation of security requirements, (3) life cycle security principles reflecting related management considerations, and (4) generic approaches to trustworthy secure system development.

V. APPLICATION IN AUTOMOTIVE ARCHITECTURE AND DISCUSSION

In this section, we show an exemplary application of our proposed security patterns to a reference architecture. Maple et al. [28] describe a reference architecture for attack surface analysis of smart cars at an abstract level which hides certain

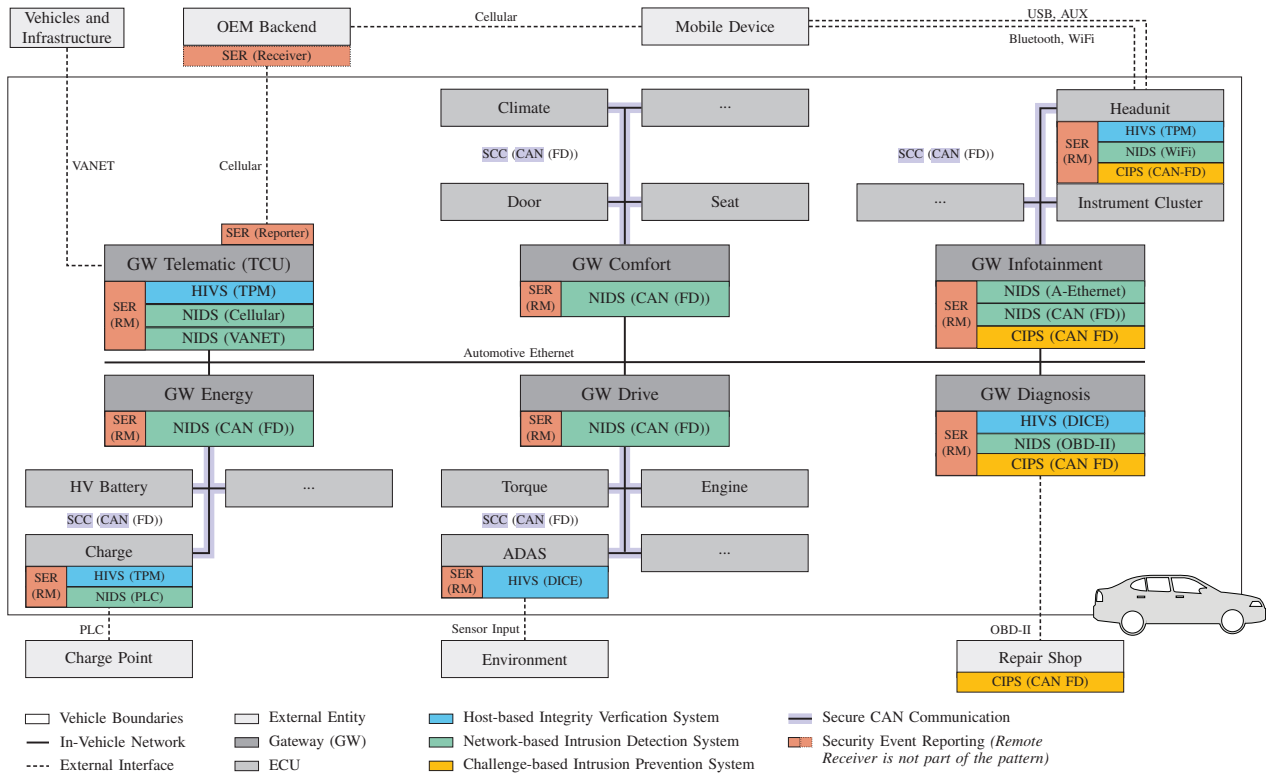

Fig. 13. Automotive Reference Architecture with proposed Observation and Mitigation Patterns.

internal interaction possibilities. In our work, we need a higher level of detail in order to show where the patterns can be used within the architecture, and thus we base our demonstration of pattern use on the reference architecture suggested in [29]. The reference architecture depicted in Fig. 13 is based on modern domain-based E/E architectures that are structured into functional domains. The resource-constraint domain subnetworks are connected via Gateway ECUs to a high-throughput backbone network, in this case based on Automotive Ethernet.

We propose to place the HIVS patterns at highly critical ECUs with external interfaces. For ECUs that offer the largest attack surface, e.g., connected to the Internet, the more secure TPM-based HIVS instantiation is chosen, while for the other ECUs the DICE-based HIVS instantiation seems sufficient.

The NIDS pattern is strategically placed to cover all communication channels in and to the outside of the vehicle.

The CIPS pattern is deployed on gateways that regulate possible cross-partition CAN messages and on the ECUs that have to know the challenge mechanism to answer correctly.

The SCC pattern is mainly placed on the bus since it aims to secure the communication on the bus by coding and decoding CAN messages according to the pattern description.

The patterns HIVS, NIDS, and CIPS make use of the SER pattern to aggregate and distribute their events. Thus, the RM component that aggregates the events needs to be present on the respective ECUs. The reporter component that sends the events to the backend needs to be placed only once within the vehicle at the TCU since this is the primary interface to the backend systems.

VI. Conclusion

Here, we have described several new patterns for augmenting high level E/E architectures to improve attack resilience.

In particular, HIVS provides a mechanism to detect anomalies (unauthorized modifications) in ECU software and firmware at boot time, NIDS provides a mechanism to detect anomalies in the in-vehicle communication, CIPS prevents unauthorized control units from successfully delivering messages, while SER provides a way to transfer security relevant events within a vehicle network to an outside SOC, and, SCC guarantees that inter ECU communications are made in a secure way.

We then presented a high-level architecture that captures the generic E/E architecture concepts currently under discussion on an abstract level. This architecture has been expanded to include possible security observation and risk reduction points as an example. The proposed patterns make it possible to identify certain steps in typical attack chains and to mitigate their consequences.

Acknowledgment

This research work has been partly funded by the German Federal Ministry of Education and Research (BMBF) and the

Hessen State Ministry for Higher Education, Research and the Arts within their joint support of the National Research Center for Applied Cybersecurity ATHENE and by the BMBF projects VITAF (ID 16KIS0835) and SAVE (ID 16KIS1324). Additionally, the project leading to this application has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 883135 (E-Corridor).

REFERENCES

[1] N. Huq, C. Gibson, and R. Vosseler. (2020) Driving Security Into Connected Cars: Threat Model and Recommendations. Accessed 2021-10-07. [Online]. Available: https://documents.trendmicro.com/assets/white_papers/wp-driving-security-into-connected-cars.pdf

[2] G. Costantino and I. Matteucci, "KOFFEE - Kia OFFensivE Exploit," Istituto di Informatica e Telematica, Tech. Rep., 2020, accessed 2021-10-07. [Online]. Available: https://sowhat.iit.cnr.it/pdf/IIT-20-2020.pdf

[3] R.-P. Weinmann and B. Schmotzle. (2020) TBONE – A zero-click exploit for Tesla MCUs. Accessed 2021-10-07. [Online]. Available: https://kunnamon.io/tbone/tbone-v1.0-redacted.pdf

[4] H. Martin, Z. Ma, C. Schmittner, B. Winkler, M. Krammer, D. Schneider, T. Amorim, G. Macher, and C. Kreiner, "Combined automotive safety and security pattern engineering approach," *Reliab. Eng. Syst. Saf.*, vol. 198, p. 106773, 2020.

[5] B. H. C. Cheng, B. Doherty, N. Polanco, and M. Pasco, "Security Patterns for Connected and Automated Automotive Systems," *Journal of Automotive Software Engineering*, vol. 1, pp. 51–77, 2020.

[6] S. Konrad, B. H. C. Cheng, L. A. Campbell, and R. Wassermann, "Using Security Patterns to Model and Analyze Security Requirements," in *2nd International Workshop on Requirements Engineering for High Assurance Systems (RHAS '03)*, C. Heitmeyer and N. Mead, Eds., 2003.

[7] SAE International, "Cybersecurity Guidebook for Cyber-Physical Vehicle Systems," SAE International, Tech. Rep. J3061, 2016.

[8] ISO/IEC, *ISO/SAE FDIS 21434 — Road vehicles — Cybersecurity engineering*, International Organization for Standardization Std., 2021.

[9] ENISA. (2019) ENISA good practices for security of Smart Cars. Accessed 2021-10-07. [Online]. Available: https://www.enisa.europa.eu/publications/smart-cars

[10] AUTOSAR, *Specification of Intrusion Detection System Protocol*, AUTOSAR Std. 981, 11 2020, accessed 2021-10-07. [Online]. Available: https://www.autosar.org/fileadmin/user_upload/standards/foundation/20-11/AUTOSAR_PRS_IntrusionDetectionSystem.pdf

[11] ——, *Specification of Intrusion Detection System Manager for Adaptive Platform*, AUTOSAR Std. 978, 11 2020, accessed 2021-10-07. [Online]. Available: https://www.autosar.org/fileadmin/user_upload/standards/adaptive/20-11/AUTOSAR_SWS_AdaptiveIntrusionDetectionSystemManager.pdf

[12] ——, *Specification of Secure Onboard Communication - CP Release 20-11*, AUTOSAR Std., 2020, accessed 2021-10-07. [Online]. Available: https://www.autosar.org/fileadmin/user_upload/standards/classic/20-11/AUTOSAR_SWS_SecureOnboardCommunication.pdf

[13] Trusted Computing Group, *TPM 2.0 Library Specification*, Trusted Computing Group Std., 11 2019, accessed 2021-07-13. [Online]. Available: https://trustedcomputinggroup.org/resource/tpm-library-specification/

[14] ——, *Hardware Requirements for a Device Identifier Composition Engine*, Trusted Computing Group Std., 03 2018, accessed 2021-07-13. [Online]. Available: https://trustedcomputinggroup.org/resource/hardware-requirements-for-a-device-identifier-composition-engine/

[17] RATS Working Group, *TPM-based Network Device Remote Integrity Verification*, Internet Engineering Task Force Std., 6 2020, accessed 2021-07-13. [Online]. Available: https://datatracker.ietf.org/doc/html/draft-ietf-rats-tpm-based-network-device-attest-00

[15] Infineon. (2019, 01) A safe for sensitive data in the car: Volkswagen relies on TPM from Infineon. Accessed 2021-07-27. [Online]. Available: https://www.infineon.com/cms/en/about-infineon/press/market-news/2019/INFATV201901-030.html

[16] A. Fuchs, H. Birkholz, I. McDonald, and C. Bormann, *Time-Based Uni-Directional Attestation*, Internet Engineering Task Force Internet-Draft draft-birkholz-rats-tuda-04, Jan. 2021, work in Progress. [Online]. Available: https://datatracker.ietf.org/doc/html/draft-birkholz-rats-tuda-04

[18] Trusted Computing Group, *Implicit Identity Based Device Attestation*, Trusted Computing Group Std., 3 2018, accessed 2021-07-13. [Online]. Available: https://trustedcomputinggroup.org/resource/implicit-identity-based-device-attestation/

[19] A. Fuchs, C. Krauß, and J. Repp, "Advanced Remote Firmware Upgrades Using TPM 2.0," in *31st IFIP International Information Security and Privacy Conference (SEC)*, J.-H. Hoepman and S. Katzenbeisser, Eds., vol. AICT-471, Ghent, Belgium, May 2016, pp. 276–289, part 7: TPM and Internet of Things. [Online]. Available: https://hal.inria.fr/hal-01369561

[20] C. Plappert, L. Jäger, and A. Fuchs, *Secure Role and Rights Management for Automotive Access and Feature Activation*. New York, NY, USA: Association for Computing Machinery, 2021, p. 227–241. [Online]. Available: https://doi.org/10.1145/3433210.3437521

[21] A. Taylor, S. Leblanc, and N. Japkowicz, "Anomaly Detection in Automobile Control Network Data with Long Short-Term Memory Networks," in *2016 IEEE International Conference on Data Science and Advanced Analytics (DSAA)*, Oct 2016, pp. 130–139.

[22] F. Fenzl, R. Rieke, Y. Chevalier, A. Dominik, and I. Kotenko, "Continuous Fields: Enhanced In-Vehicle Anomaly Detection using Machine Learning Models," *Simulation Modelling Practice and Theory*, vol. 105, p. 102143, 2020. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1569190X20300824

[23] Y. Chevalier, R. Rieke, F. Fenzl, A. Chechulin, and I. Kotenko, "ECU-Secure: Characteristic Functions for In-Vehicle Intrusion Detection," in *International Symposium on Intelligent and Distributed Computing*. Springer, 2019, pp. 495–504.

[24] 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.

[25] G. Bella, P. Biondi, G. Costantino, and I. Matteucci, "CINNAMON: A Module for AUTOSAR Secure Onboard Communication," in *16th European Dependable Computing Conference, EDCC 2020, Munich, Germany, September 7-10, 2020*. IEEE, 2020, pp. 103–110.

[26] E. Metzker. (2020) Reliably Detecting and Defending Against Attacks - Requirements for Automotive Intrusion Detection Systems. VECTOR.

[27] J. C. O. Ron Ross, Michael McEvilley, "Systems Security Engineering: Considerations for a Multidisciplinary Approach in the Engineering of Trustworthy Secure Systems," U.S. Department of Commerce, Washington, D.C., Tech. Rep. NIST Special Publication 800-160, Volume 1, 2018.

[28] C. Maple, M. Bradbury, A. T. Le, and K. Ghirardello, "A Connected and Autonomous Vehicle Reference Architecture for Attack Surface Analysis," *Applied Sciences*, vol. 9, no. 23, 2019.

[29] C. Plappert, D. Zelle, H. Gadacz, R. Rieke, D. Scheuermann, and C. Krauß, "Attack Surface Assessment for Cybersecurity Engineering in the Automotive Domain," in *2021 29th Euromicro International Conference on Parallel, Distributed and Network-Based Processing (PDP)*, 2021, pp. 266–275.