DIAS

Smart Adaptive Remote Diagnostic Antitampering Systems

EUROPEAN COMMISSION

HORIZON 2020

LC-MG-1-4-2018

Grant agreement ID: 814951

Deliverable No.	D4.4
Deliverable Title	Validation and verification methodology and results
Issue Date	31/10/2022
Dissemination level	Public
Main Author(s)	Miao Zhang, FEV.io GmbH
	Shaoning Chen, FEV.io GmbH
	Gianmarco Baldini, JRC
	Björn Siegel, Robert Bosch GmbH
Version	V1.0

DIAS Consortium

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 814951.

Document log

Version	Description	Distributed for	Assigned to	Date
V0.1	Initial ToC for Review	review	FEV	7/12/2021
V0.2	Added content from subcontract to University of Murcia	review	JRC	15/02/2022
V0.3 - V0.5	Draft content of methodology part	Content review	UMFST, BOSCH, CERTH	13/06/2022
V0.6 - V0.8	Partners input and draft content of complete deliverable	Content review	UMFST, CERTH, BOSCH, FORD	19/09/2022
V0.9	Reviewer's comments addressed	GA check	GA members	12/10/2022
V1.0	First final version	-	-	31/10/2022

Verification and approval of final version

Description	Name	Date
Verification of the "Final content of deliverable (v0.9)" by WP leader	Miao Zhang	24/10/2022
Check of the "First final version (v1.0)" before uploading by coordinator	Zissis Samaras	16/11/2022

Executive summary

Cybersecurity testing of the vehicles has received increased level of attention from the regulatory, industry and research community in recent years because of the increase level of connectivity and computing complexity of modern vehicles, which can make them vulnerable to attacks and manipulation. One specific aspect investigated in DIAS is the potential for tampering of the vehicle and its internal components (e.g., Electronic Control Units, in-vehicle network and sensors) because economical or performance benefits can be gained by tampering with the regulations (e.g., emission levels can be modified to improve vehicle performance at the cost of increasing pollution). The research area of testing modern vehicles regarding this aspect is still in early phases but the significant efforts by the research, industry, regulatory and standardization communities from vehicle cybersecurity testing can be reused because of the many similarities. This deliverable provides a multi-pronged analysis of the problem by: 1) conducting an evaluation of the potential methodologies, which can be applied to this context, 2) reporting on the testing activities of the DIAS demonstrator developed to address potential tampering risks, 3) providing a concept review of one proposed security framework, which complements the DIAS demonstrator.

More in detail, the first part of this deliverable provides an overview of the potential methodologies already defined in the other domains for validation and verification including risk assessment methodologies in this specific context. The specific aspects of modern vehicles are identified and compared to assess which methodologies are most appropriate for the automotive context.

Then, the second part of this deliverable reports and describes the activities of cybersecurity and anti-tampering testing implemented by WP4 partners on the technical solutions developed by the DIAS project and consortium partners to improve the robustness of modern vehicles against tampering. Based on the security analysis and market assessment of tampering devices performed in earlier tasks within DIAS, three categories of the security are defined as critical:

- Category 1 is the Engine Control Unit (ECU) reprogramming.
- Category 2 is the CAN-bus communication between the ECU and the sensors.
- Category 3 is the data sending to the cloud.

In Task 4.2 and 4.3, several security solutions are developed covering all the three categories to ensure the security of the whole system. In this part of the deliverable, penetration testing activities targeted for the security of Category 1 and 2 are documented and the results are reported. The Category 3 was verified via design and code review by an external company, and the results will be reported in a separate document.

The third part of this deliverable provides the concept review performed by the DIAS partners on the security framework designed to complement the DIAS demonstrator. The deployment aspects including the potential complexity or scalability of the implementation and deployment of the approach are also discussed.

Contents

List of Abbreviations

C2C-CC	Car 2 Car Communication Consortium
ABS	Anti-lock Braking System
AC	Access Control
ACLs	Access Control Lists
ASIL	Automotive Safety Integrity Level
CAN	Controller Area Network
CC	Common Criteria
CCRA	Common Criteria Recognition Arrangement
CERT	Computer Emergency Response Team
CHASSIS	Combined Harm Assessment of Safety and Security for Information Systems
cPP	collaborative Protection Profiles
CSMS	Cybersecurity management systems
CVC	Code Verification Certificate
CVSS	Common Vulnerability Scoring System
CWRAF	Common Weakness Risk Analysis Framework
CWSS	Common Weakness Scoring System
DEF	Diesel Exhaust Fluid
DoW	Description of Work
DSL	Domain Specific Languages
DTC	Diagnostic Trouble Code
EAL	Evaluation Assurance Levels
ECC	Elliptic Curve Cryptography
ECDSA	Elliptic Curve Digital Signature Algorithm
ECSO	European Cyber Security Organisation
ECU	Engine Control Units
ENISA	European Union Agency for Cybersecurity
EPS	Environmental Protection System
ETSI	European Telecommunication Standard Institute
EVITA	E-safety Vehicle InTrusion protected Applications
FACT	Failure Attack and Countermeasure
FMVEA	Failure Mode Vulnerabilities and Effect Analysis
GCU	General Control Unit
GNSS	Global Navigation Satellite Systems
GPS	Global Positioning System
HARA	Hazard Analysis and Risk Assessment method
HARM	Hailstorm Application Risk Metric
HEAVENS	HEAling Vulnerabilities to ENhance Software Security and Safety
HSM	Hardware Secure Module
IETF	Internet Engineering Task Force
ISO	International Standardization Organization
JSON	JavaScript Object Notation
KMS	Key Management Solution
LIN	Local Interconnect Network
MAC	Message Authentication Code
MITM	Man-in-the-middle

MBT	Model-Based Testing
MoRA	Modular Risk Assessment
MOST	Media Oriented Systems Transport
MUD	Manufacturer Usage Description
NIST	National Institute of Standards and Technology
NVD	National Vulnerability Database
OBD	On-board Diagnostics.
OCTAVE	Operationally Critical Threat, Asset, and Vulnerability Evaluation
OEM	Original equipment manufacturer
OTA	Over The Air
PKS	Production Key Server
PP	Protection Profiles
PRF	Pseudo Random Function
QM	Quality Management
SAE	Society of Automotive Engineers
SAST	Static Application Security Testing
SCA	Static Code Analysis
SCU	Sensor Control Unit
SecOC	Secure Onboard Communication
SecOC Light	lightweight Secure Onboard Communication
SEI	Software Engineering Institute
ST	Security Target
SUMS	Software Updates Management System
SUT	System Under Test
TAL	Trust Assurance Levels
TARA	Threat analysis and risk assessment methods
TOE	Target Of Evaluation
TVRA	Threat, Vulnerability, and Risk Analysis
UDP	User Datagram Protocol
UDS	Unified Diagnostic Services
UML	Unified Modelling Language
UNECE	United Nations Economic Commission for Europe
xCU	Electronic Control Unit, Any Control Unit
YANG	Yet Another Next Generation

List of Definitions

AdBlue™/DEF: an aqueous urea solution made with 32.5% urea and 67.5% deionized water. DEF is consumed in SCR that lowers nitrogen oxides (NOx) concentration in the diesel exhaust emissions from a diesel engine.

Attack vector: Approach, or the sequence of steps used by an attacker to gain access to the target system.

Attacker: An individual who attempts to access the vehicle's network, mostly without the owner's consent.

Authentication: Verifying the identity of a person or a communication partner.

Data authentication: The process of verifying the origin and integrity of data.

Data integrity: The receiver of data must have the assurance that the data has come intact from the intended sender and not changed intentionally and unintentionally.

ECU (Engine Control Unit): ECU is a type of electronic control unit that controls a series of actuators on an internal combustion engine to ensure optimal engine performance.

ECU reprogramming or flashing: electrical operation of reprogramming an ECU memory.

Emulator: a device intended to take over control of a tampered EPS.

EPS: includes all systems on a vehicle that reduce exhaust emissions. The current context includes SCR, DPF, EGR and TWC systems.

OBD (system): a system that continually monitors the electronic sensors of engine and EPS subsystems. When a potential problem is detected, a dashboard warning light (MIL) is illuminated to alert the driver.

OEM or Manufacturer: Person or body that makes goods for sale. With regard to vehicle manufacturing and especially environmental protection systems, OEM is a person or body who is responsible to the approval authority for all aspects of the type-approval or authorization process and for ensuring conformity of production. The person or body doesn't have to be directly involved in all stages of the construction of the vehicle, system, component, or separate technical unit, which is the subject of the approval process, as defined in directive 2007/46/EC.

Tamperer: A person who intentionally, illegally and for whatever reason alters an EPS, resulting in increased emissions.

xCU: Used to refer to the different Electronic Control Units of a vehicle, where "x" stands for whichever of the electronic control unit modules (e.g. **Powertrain** Control Unit – **P**CM, **Transmission** Control Unit – **T**CM etc.)

List of Figures

List of Tables

1 Introduction

1.1 Background

A report from Juniper Research[1] revealed that by the year 2023 around 775 million consumer vehicles will be connected via telematics or by in-vehicle apps, rising from 330 million vehicles in 2018. This represents an average annual growth of 18.7% in 5 years. In parallel to the great growth that this industry is having, especially in terms of connectivity, there is growing concern about the cybersecurity offered by these systems. Smart vehicles are not only exposed to specific risks in this context, but some of them may involve the loss of human life. Therefore, the industry needs to adopt a strict low risk tolerance and manage the security throughout the lifecycle of the smart vehicle.

One of the vehicles cybersecurity efforts can be observed through the regulations developed within the United Nations Economic Commission for Europe (UNECE). In particular, the new UNECE regulations UN 155 requires to manage the vehicle cyber risks, validate the security of the vehicle from the design phase, detect and mitigate incidents and provide safe and secure software updates.

Beyond the specific aspects of cybersecurity in vehicles, tampering is growing in importance. Tampering is mostly related to circumvent or make the vehicle not compliant to regulations for a variety of reasons including:

- *economic gains*. The entity performing the tampering activity can profit from tampering because he/she can be able to avoid the regulatory constraints at the expenses of law abiding parties or at the expenses of the safety on the road. For example, a tamperer may not be compliant to the on-board weighing Regulation (EU) 2019/1213 to carry more weight (and thus raising the profit for each trip) than what it should be permitted. In another example, related to the tachograph application and regulation Regulation (EU) No 165/2014, the tamperer can alter the recorded driving time. In both cases, the tamperer has an economical gain because he(she) can carry more weight or drive more hours thus increasing the profit at the risk of driver fatigue or safety to the other parties in the road infrastructure.
- *performance gains* to increase the performance of the vehicle or its use. For example, the tamperer can alter the vehicle emissions to improve the performance (e.g., power) of the vehicle.
- *Competitive advantage.* This may be legated to economic gains, but it may also provide other benefits to the tamperer by establishing a competitive advantage in comparison to other competitors in the market. For example, an increase of driving hours in the tachograph regulation may put competitors out of the market.

Tampering activities may be closely related to cybersecurity but not necessarily because tampering can also be implemented by mechanical or chemical means. On the other side, as modern vehicles are becoming more digital and dependent on computing systems for implementation and monitoring, cybersecurity attacks can be used to implement tampering. For example, the software of the computing platform used to collect the driving records can be altered using a cybersecurity attack.

More in general, the methodologies used to identify and assess the risks of tampering has many similarities with the methodologies used to identify cybersecurity risks. However, currently the standards for managing the risk of smart vehicles have shortcomings that make it difficult to meet these requirements. One of the main challenges is the evaluation of the risk of the vehicle considering not only safety, but also security. Before 2022, vehicular standards related to the digital aspects of

[1] https://www.juniperresearch.com/press/in-vehicle-commerce-opportunities-exceed-775mn

vehicles were predominantly focused on safety aspects. Recently, the ISO 21434 standard has become de-facto worldwide standard which clearly tell how security risks shall be evaluated in the vehicular domain. It should also be noted that security failures can lead to safety failures, so these two concepts cannot be treated independently. This is coupled with the fact that the automotive environment has unique characteristics, such as the high interdependence of components, which makes security assessment even more difficult, requiring mechanisms and tools to manage cascade effects and dependencies. Furthermore, it is hard to define a common standard methodology to describe how security evaluation and certification must be done. The wide variety and heterogeneity of methodologies, mechanisms, standards and components derives on a confusing landscape of solutions. Therefore, it is quite unclear which security aspects should be considered to guarantee an adequate security level. In this context, comparability is unfeasible, as different schemes use their own metrics, especially when vehicles are evaluated under different national schemes or approaches, or when they include some subjective or difficult to calculate metrics (e.g., likelihood).

Then, there is the need to define a methodology for the cybersecurity testing and evaluation of modern vehicles, which can detect and counter the security vulnerabilities introduced during software development and integration but also when cybersecurity vulnerabilities are identified in the lifetime of the vehicle and vehicle type.

In particular, the mitigation of tampering threats is the main focus of the Horizon 2020 DIAS project. In this context, tampering is a specific type of attack which aims to modify the behaviour of an internal automotive function in such a way that the tamperer (i.e., the malicious party) has a benefit either economical or by performance. For example, the vehicle can move faster because a control on emission limits has been removed or it consumes less material (e.g., fuel, AdBlue) because of the tampering or because the odometer is manipulated to show false values in the km counter, which can be exploited when the vehicle is sold.

Then, there is slight difference between a classical vehicle cybersecurity attack where the goal is to eventually ‘disrupt’ an automotive function and the more subtle tampering action to gain an advantage usually against a regulation. On the other side, the overall methodology to identify the main risks of tampering can be also identified using a similar methodology.

The proposed methodology combines risk assessment and testing in such a way that the security assessment can be carried out in a more objective and empirical way, also favouring the automation of the security and anti-tampering testing processes. Furthermore, the proposed methodology considers safety as an important aspect, so that these two concepts (security and safety) are taken into account together, and not independently.

1.2 Purpose of the document

The purpose of the document is to describe approaches for verification and validation of modern vehicles in relation to tampering. Such approaches are validated on the results of security testing of the DIAS demonstrator developed in the DIAS project. More in detail, this document reports on:

1) the analysis and evaluation of the potential methodologies, which can be applied to the context of risk management, tampering and security,
2) on the testing activities of the DIAS demonstrator developed to address potential tampering risks,
3) providing a conceptual review of the proposed security framework, which complements the DIAS demonstrator.

1.3 Document structure

The document structure is following:

- Chapter 2 provides the overall context on automotive cybersecurity with the identification of the current stakeholders and regulatory framework. This chapter is needed to introduce the context and as a reference for the subsequent chapter.
- Chapter 3 describes the proposed security evaluation methodology for modern vehicles including the risk management and the concept of certification of modern vehicles. This chapter provides an overview of the main risk management approaches and how they can be applied to this context of cybersecurity validation and verification. Then, the link to the testing approach is defined with relation to relevant standards.
- Chapter 4 describes the methodology for security testing in modern vehicles. Firstly, an overview on cybersecurity testing methods and processes is provided including penetration testing, fuzzy testing and so on. Then it is described, how these generic testing methodologies can be applied to the cybersecurity of modern vehicles. In this context, an overview of the related standards for cybersecurity testing is provided. Finally, the chapter describes how to address the software lifecycle and the related testing and evaluation of software updates and patches.
- Chapter 5 reports on security testing for the validation and verification of the DIAS demonstrator for specific vulnerabilities.
- Chapter 6 provides the concept review analysis.
- Chapter 7 concludes the report.

1.4 Deviations from original DoW

1.4.1 Description of work related to deliverable as given in DoW

Validation and verification methodology and results: approaches for verification and validation will be documented, the results of security testing will be described.

1.4.2 Time deviations from original DoW

None reported.

1.4.3 Content deviations from original DoW

None reported.

2 Automotive cybersecurity context

The current automotive cybersecurity context is changing in modern times, with automotive vehicles becoming increasingly sophisticated with an increasing number of sensors, computing platforms and connectivity. This trend paves the way for the more sophisticated automated vehicles of the future where some or all functions can be delegated from the human driver to the vehicle itself. For this reason, modern automotive vehicles are often called “computer on wheels” or smart vehicles or smart cars.

Following the definition of ENISA [1], smart cars are systems providing connected, added-value features in order to enhance car users’ experience or improve car safety. It encompasses use cases such as telematics, connected infotainment or intra-vehicular communication.

Smart cars can be classified according to its level of automation. In particular, the Society of Automotive Engineers (SAE) [2] defines six levels of driving automation for on-road vehicles:

- Level 0, No automation: fully manual vehicle, as the driver performs all driving tasks.
- Level 1, Driver assistance: the vehicle is still controlled by the driver, but the vehicle includes some assistant features (e.g., steering, speed or breaking control).
- Level 2, Partial automation: the vehicle automates certain parts of the driving experience (e.g., self-parking features), but the driver remains in complete control of the vehicle.
- Level 3, Conditional automation: the driver is not required to monitor the environment, but human override is required when the machine is unable to execute the task at hand or the system fails. It is the lowest level considered as an automated driving system.
- Level 4, High automation: the vehicle is capable to intervene itself under certain conditions if things go wrong or if there is a system failure. The driver also has the possibility of controlling the vehicle.
- Level 5, Full automation: the vehicle is capable to intervene itself under all conditions if things go wrong or if there is a system failure. The driver also has the possibility of controlling the vehicle.

2.1 Architecture and components

The purpose of this chapter is to describe the typical architecture of smart vehicles and the main assets or components associated to them, which will be key when evaluating the security of the whole system. Most car architectures distinguish between different interconnected domains (subsystems), each of them corresponding to different features of the car [3]. In particular, Figure 1, distinguish five different domains and one sub-domain:

- Infotainment (media, radio, navigation, phone, GNSS, etc.)
- Driveability (engine control, transmission).
- Chassis (lights, wheels, steering control, anti-lock braking system (ABS), etc.)
- Comfort (door locking, climate control, seat belt, airbags, etc.)
- Diagnosis (external systems interfaced with the car)

Whereas Figure 1 shows a particular topology of an in-vehicle network, it is worth noting that other topologies exist, e.g., bus topology, star topology, ring topology and mesh topology. Moreover, other network topologies (hybrid topologies) can be created by combining these basic topologies [4].

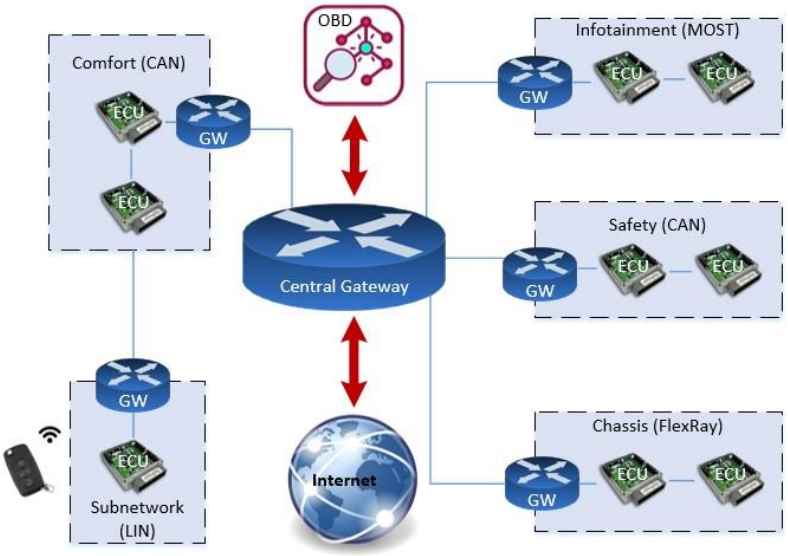


Figure 1 High level model of the in-vehicle architecture

Each domain is comprised of one or several electronic control units. The term xCU is also used in this deliverable to indicate control unit components which are not specifically electronic control units. An electronic control units can vary from a standard switch point to a component that controls the anti-lock braking system. An electronic control unit is what we see as either a communication node or subsystem node within a very compact, yet complex network infrastructure. Although the electronic control unit is concerned with everything inside the vehicle network, outside the vehicle, the communication unit (or CU) takes care of communication. All information from the user or Original Equipment Manufacturer (OEM) backend in the vehicle will pass through this CU to reach the vehicle's internal network [5].

One of these special domains is the On-board Diagnostics (OBD). It acts as a typical gateway to access diagnostic information of the entire vehicle network infrastructure.

In this scenario, cross-system functions require coordination between the different systems, which exchange a large amount of data. In addition to powerful components, a powerful communication system and a low-cost network suitable for automobiles are also required. For this purpose, the architecture also considers the usage of a special serial data bus system. The usage of a bus in comparison to a solution that uses conventional wiring, reduces costs (fewer cables), has a better reliability (fewer plug-in connections), simplifies the vehicle assembly during production, allows multiple sensor signals and provides an easier handling of equipment variants within the vehicle [4].

Because of differing requirements, bus systems can be subdivided into the following classes:

- Class A, whose representative is the Local Interconnect Network (LIN) bus. This class has low data rates (up to 10kBits/s) and it is suitable for actuator and sensor networking.
- Class B, whose representative is the low speed controller area network (CAN), with average data rates of 125 kBits/s and suitable for complex mechanisms for error handling and control unit networking in the comfort functions.
- Class C, whose representative is the high speed CAN (1Mbit/s), useful for real time requirements and control unit networking in the drive and running gear functions.
- Class C+, whose representative is 8the FlexRay bus, with high data rates 10MBits/s) that makes it also suitable for real time requirements and control unit networking in the drive and running gear functions.

- Class D, whose representative is the Media Oriented Systems Transport (MOST) bus, with very high data rates (more than 10MBits/s), suitable for control unit networking in the telematics and multimedia functions.

The CAN bus is the most popular and standardised bus used today. This serial bus protocol is mainly used in the safety and comfort subsystems of the in-vehicle network, and sometimes also used in the chassis subsystem. All CAN nodes have the ability to transmit data, and multiple CAN nodes can simultaneously request the bus. With its real-time characteristics and small data packet format, CAN can be implemented as a low-cost vehicle network. However, these attributes also make such networks vulnerable to attacks [5].

Because CAN bus packets are broadcast, all controllers on the same network see every packet, kind of like User Datagram Protocol (UDP) on Ethernet networks. The packets don't carry information about which controller (or attacker) sent what. Because any device can see and transmit packets, it's trivial for any device on the bus to simulate any other device. Therefore, an attacker could exploit the CAN bus to for example install a malicious diagnostic device to send packets to the CAN bus, plug directly in to a CAN bus to attempt to start a vehicle without a key or to upload malware, install a malicious diagnostic device to track the vehicle or even to install a malicious diagnostic device to enable remote communications directly to the CAN bus, making a normally internal attack now an external threat [6][7].

2.2 Interfaces

To access to the different components, the vehicle offers different interfaces that we can categorise in infotainment, OBD and others. These interfaces represent the entry points an attacker can use to perform an attack over the vehicle [8].

The infotainment system offers more attack surfaces than any other vehicle component (auxiliary jack, internal network control and wireless inputs such as GNSS, Bluetooth or radio). Gaining access to the infotainment system will not only allow you to modify the infotainment itself but also will open a door to additional information about how your vehicle works, such as how it routes CAN bus packets and updates the electronic control unit. As an example, an attacker could use the infotainment interface to put the console into debug mode, to alter diagnostic settings, find an input bug that causes unexpected results, install malware to the console, use a malicious application to access the internal CAN bus network or to eavesdrop on actions taken by vehicle and even use a malicious application to spoof data displayed to the user, such as the vehicle location [6].

Although initially, the OBD port requires physical access to obtain diagnostic information from a vehicle, the widespread use of Wi-Fi, cellular networks, and Bluetooth connectivity, has become in the norm the remote access the diagnostic ports. Due to the wireless accessibility that is now available with connected vehicles, so comes the risk from cyberattack from a range of malicious actors, for example to demand payment for the repair of a hacked vehicle or to access to the CAN bus and to the individual electronic control units, which are connected to the OBD [8].

Finally, other interfaces such as those used for firmware upgrades, externals sensors or cloud providers can be also entry point for an attack, forcing the vehicle to perform unwanted actions or resulting in backdoor attacks linking the automobile to the attacker's system [9],[10].

2.3 Stakeholders

From the United Nations Economic Commission for Europe (UNECE) regulation [11] and the support documents from ENISA [12][13], we can identify the different stakeholders in the security management process within smart vehicles:

- The Approval Authorities, in charge of granting type approval with regard to cybersecurity to such vehicle types that satisfy the requirements of the UNECE Regulation.
- The Technical Service, in charge of verifying by testing of a vehicle that the vehicle manufacturer has implemented the cybersecurity measures they have documented. Tests shall be performed in collaboration with the vehicle manufacturer.
- The vehicle manufacturer, which shall demonstrate to an Approval Authority or Technical Service that their cybersecurity Management System applies to the development, production and post-production phases, and satisfies the basic considered in the UNECE regulation.
- Tier-1 and Tier-2 suppliers. Vehicle manufacturing is a heavily tiered ecosystem, and manufacturers integrate components provided by suppliers, which are labelled as “Tier-1”. While Tier-1 suppliers have direct contractual relationships with car manufacturers to provide car components, the ecosystem also includes suppliers labelled as “Tier-2”, which only have contractual relationships with Tier-1 suppliers (e.g., for plastics, mechanical parts, molds, electronic components or software).
- Aftermarket suppliers. Users can also buy aftermarket products from other vendors; for example, smart dongles used on the OBD-II port, providing additional features to their car. More traditional aftermarket products may include media players or third-party GNSS.
- Policy makers, in charge of developing the regulation that establishes the minimum security level required for smart vehicles and the procedures the manufacturer should follow to guarantee such requirements.
- Users, which are the drivers of the vehicle.

2.4 Current Regulations

Currently, the most important regulations regarding smart vehicles are the UN Regulation on uniform provisions concerning the approval of vehicles with regards to cybersecurity and cybersecurity management system [11] and the UN Regulation on uniform provisions concerning the approval of vehicles with regards to software update and software updates management system (SUMS) [14]. Indeed, they have been accepted and will be required to sell vehicles by July 2022.

While, cybersecurity is not exactly tampering, cybersecurity threats can be used to implement tampering. In addition, The UNECE regulation for type approval is the most important regulation for modern vehicles and actions to circumvent or avoid the implementation of the regulation is a tampering action. For this reason, this section is provided in this deliverable.

As reported by UNECE, *the two new UN Regulations (…) require that measures are implemented across 4 distinct disciplines: Managing vehicle cyber risks; Securing vehicles by design to mitigate risks along the value chain; Detecting and responding to security incidents across vehicle fleet; Providing safe and secure software updates and ensuring vehicle safety is not compromised, introducing a legal basis for so-called “Over-the-Air” (O.T.A.) updates to on-board vehicle software*. In particular, whereas the first regulation focuses on the unified provisions for vehicle approval related to cybersecurity and cybersecurity management systems (CSMS), the second one focuses on the vehicle software update process and the software update management system.

Although the regulation specifically states what needs to be done (i.e., the high level requirements), it does not specifically define how to meet such regulatory requirements, nor does it specify detailed technical measures that the vehicle manufacturers are obliged to implement. The manufacturers can fulfil the requirements defined in the regulation (i.e., the 'what') by using manufacturer specific implementations or standards (i.e., the 'how'), granted that the requirements are validated in the testing phase (e.g., type approval). The explanation of this approach is that using relevant standards (such as ISO/SAE 21434) and implementing appropriate mitigation measures, OEMs should be able to demonstrate if the principles of this regulation are met.

It is worth noting that the regulation considers a process of certification and approval, demanding evidence based on test reports and threat modelling, in order to prove that cybersecurity is ensured throughout the lifecycle of the vehicle.

Finally, in Annex 5, the regulation includes a list of cybersecurity threats and their corresponding mitigations to secure a vehicle, its components, and back-end servers against these threats. However, although the list of threats, vulnerabilities, and mitigations is extensive, it is not exhaustive and should be complemented.

3 Security and Anti-Tampering Evaluation Methodology

3.1 Introduction

This chapter describes the proposed security and anti-tampering evaluation methodology for modern vehicles including the risk management and the concept of certification of modern vehicles. This chapter provides an overview of the main risk management approaches and how they can be applied to this context of cybersecurity validation and verification. Then, the link to the testing approach is defined with relation to relevant standards.

3.2 Overall methodology

As mentioned in the UNECE regulation, the processes used within the manufacturer's organization to manage cybersecurity should include:

- Processes of risk identification to vehicle types. Within these processes, the threats collected in the UNECE document, and other relevant threats shall be considered.
- Processes for assessment, categorization and treatment of the risks identified.
- Processes used to verify that the risks identified are appropriately managed.
- Processes used for testing the cyber security of a vehicle type.
- Processes for ensuring that the risk assessment is kept current.
- Processes used to monitor for, detect and respond to cyber-attacks, cyber threats and vulnerabilities on vehicle types and the processes used to assess whether the cyber security measures implemented are still effective in the light of new cyber threats and vulnerabilities that have been identified.
- Processes used to provide relevant data to support analysis of attempted or successful cyber-attacks.

The proposed methodology for security evaluation explicitly considers these processes and deals with some of the challenges within the automotive domain analysed in the previous chapter. In particular, the methodology follows a test- based risk assessment approach, in which the risk assessment is improved by a security testing process.

As described in the next subchapter 3.3, risk assessment includes the risk identification and risk estimation phases described in this chapter.

Beyond UNECE, security assessment and testing have been already considered in current literature as essential processes for cybersecurity evaluation. Indeed, ECSO considers them as key elements for cybersecurity certification: " *... It would be convenient to consider a security testing methodology (to) help in [...] the process of updating the certificate in a fast, easy and inexpensive manner. When doing an update or patch, security tests can be executed to assist reassessment processes...*" [62]. ETSI [63] also considers the combination of risk assessment and testing defining two different approaches based on the ISO 31000 standard for Risk Management and the ISO 29119 standard for Security Testing.

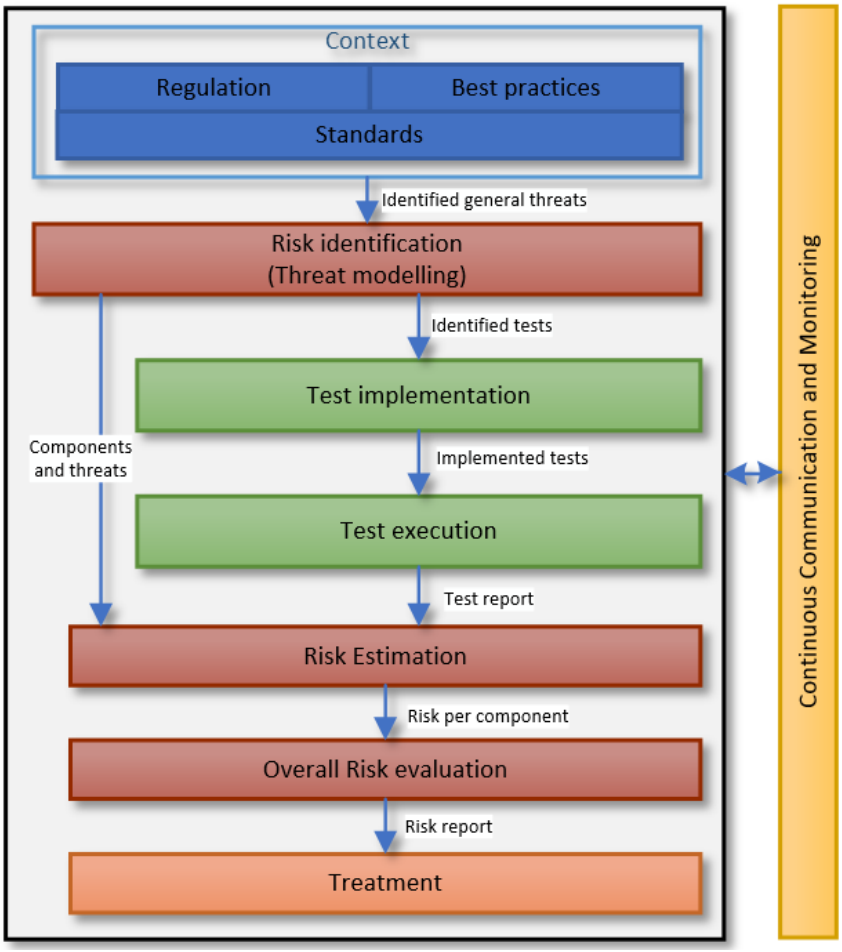


Figure 2 Potential methodology

In the same way, the proposed security evaluation methodology builds a framework on top of the two main streams of this proposal: security testing to identify security vulnerabilities, and security risk assessment to measure the associated risk. Figure 2 shows the high level view of the proposed methodology, distinguishing several processes. The first one, is the context phase, which takes into account the existing regulation in smart vehicles, the best practices, current standards, etc. to build an initial set of security claims that can be used as starting point for the security evaluation. From this initial set, and taking into account the particular Target of Evaluation (TOE), in the risk identification phase, we select a set of applicable threats. This set can be also extended with specific threats not considered in the initial set, by examining the special characteristics of the TOE. Once the threats are selected, the test implementation phase deals with the design and implementation of the tests necessary to verify if the system is vulnerable to these threats. The tests are executed in the test execution phase, generating at the end of the process a test report. The test results of the previous phase are used to estimate the risk of every component of the TOE during the risk estimation phase. Towards this end, information from the risk identification phase is required, regarding the components, the identified threats and their impact. The overall risk evaluation phase combines the risk coming from every component, obtaining an overall measure of the system security. At the end of the evaluation process, we generate a report with the evaluation results. Other actions are also possible to mitigate the security flaws encountered during the process. Additionally, the methodology also considers a transversal and supportive process for continuous communication and monitoring meant to deal with the lifecycle management of the TOE.

We highlight that the methodology shown in Figure 2 does not seem to be the classical V methodology for validation and verification as in ISO/SAE 21434:2021, whose V methodology is reported here in Figure 3. On the other side, each of the tasks defined in Figure 2 can be associated to the tasks defined in the ISO/SAE 21434 V model. The advantage of using the methodology described in Figure 2 is the more specific focus on the risk assessment and the identification of the main risks, which can change in time depending on new regulations, technologies advancements in tampering tools or more experienced malicious entities involved in tampering. Such an approach based on high level risk identification is also proposed in UNECE UN 155 [71]. On the other side, ISO/SAE 21434:2021 is more specifically linked to the production lifecycle (e.g., production of the anti-tampering solutions) and it can describe well this specific part of the overall methodology.

For example, the security goals identified in the ISO 21434 are the first step in the methodology of Figure 2 while the testing activities and implementation are similar.

In both cases, it is an iterative process, where the cybersecurity process is continuously iterated version after version.

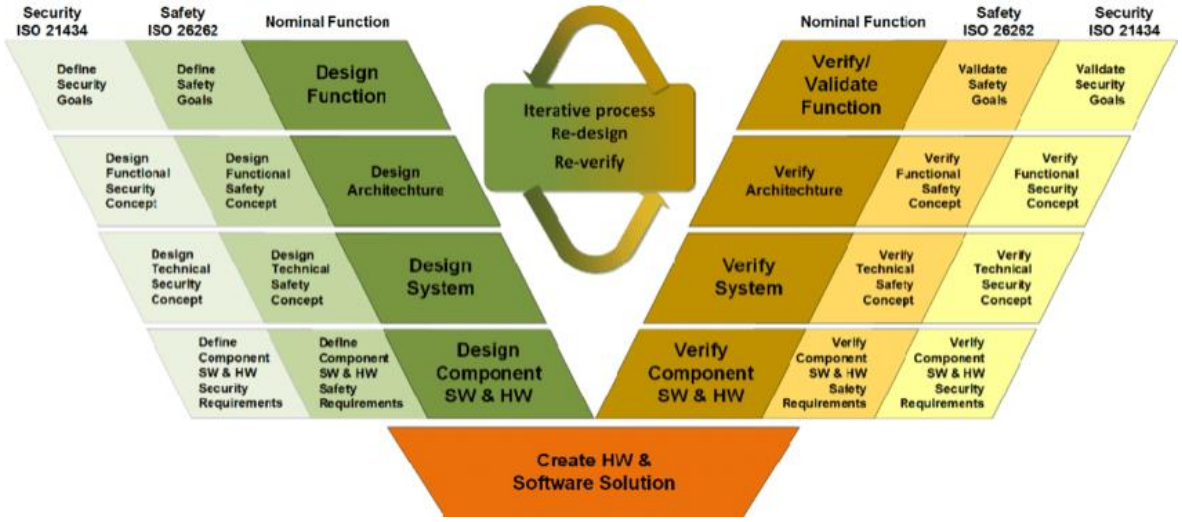


Figure 3 The ISO/SAE 21434 standard maps safety and security requirements into the traditional systems engineering V model to minimize the effort associated with designing safe and secure automotive systems (source https://www.embeddedcomputing.com/application/automotive/isosae-21434-a-joint-solution-to-the-automotive-cybersecurity-challenge)

A mapping of the steps from the methodology described in Figure 2 and ISO/SAE 21434 is shown in the following Table 1. We would like to highlight that this mapping with ISO/SAE 21434 is just an example. A similar mapping can be done with other standards.

Table 1 Mapping between the proposed methodology and ISO/SAE 21434

Proposed Methodology	ISO/SAE 21434
Context	Concept phase (section 9 of ISO/SAE 21434)
Identification of general threats	Threat analysis and risk assessment methods (TARA) (section 15 of ISO/SAE 21434)
Risk Identification	Part of the TARA. For examples sections 15.3-15.7 of ISO/SAE 21434.

Test Implement and Execution	Integration and Verification. (section 10.4.2 of ISO/SAE 21434)
Risk Estimation	Risk Value Determination (section 15.8 of ISO/SAE 21434)
Overall Risk Validation	Cybersecurity Validation (section 11 of ISO/SAE 21434)
Treatment	Risk Treatment decision (section 15.9 of ISO/SAE 21434)
Continuous communication and monitoring	Continual Cybersecurity Activities (section 8.3 of ISO/SAE 21434), Cybersecurity Monitoring (section 8.4, 8.5 of ISO/SAE 21434.

3.2.1 Definition of the context

As already mentioned, the security evaluation proposal intends to evaluate the security level of a certain TOE. Although the TOE is defined by Common Criteria (CC) as a set of software, firmware and/or hardware possibly accompanied by guidance, we also consider its configuration (i.e., a specific protocol, libraries, cryptographic parameters, etc.) as part of the TOE, as well as the context in which it is intended to operate.

Also, in order to evaluate the security of a system, a starting point is necessary, through which to analyse possible security flaws. In general, risk assessment schemes use existing vulnerabilities for this, which are collected in databases such as the well-known National Vulnerability Database (NVD) [72], as well as the security expert analysis. The starting point of the methodology defined in this deliverable in set up in the Context phase. Here we do not only consider known threats from vulnerability databases, but we also consider best practices and standards that could be used to create a set of requirements to protect the system against unknown threats. It is also worth noting that specially in a sensitive domain such as smart vehicles in which people life is involved, it is crucial to comply with the current regulation to guarantee an acceptable quality regarding safety and security. Some examples of sources include standards ETSI TS 103 096 to ETSI TS 103 096, UNECE and GDPR regulation, ISO 26262, ISO/SAE DIS 21434, SAE J3061 and best practices from ENISA [13][59].

Furthermore, this initial and general set can be used in a certification process to define a set of profiles associated to the level of testing performed or the level of verification of the requirements, similarly to the Evaluation Assurance Levels (EALs) defined in the Cybersecurity Act, in Common Criteria [104][105], the Trust Assurance Levels (TAL) defined by the Car 2 Car Communication Consortium (C2C-CC) [106], the ENISA specifications [107], the security levels of the HEAVENS approach or the Automotive Safety Integrity Level (ASIL) levels defined within the ISO 26262 standard, as shown in Table 2.

Table 2 Mapping between different approaches for security levels. QM stands for Quality Management.

CC	ISO 26262	HEAVENS	C2C-CC:TAL
0	QM	QM	0
EAL1	ASIL A	Low	1
EAL2	ASIL A	Low	2
EAL3	ASIL B	Medium	3
EAL4	ASIL C	High	4
EAL4+	ASIL D	Critical	5

3.2.2 Risk Identification

Once we have set up an initial set of requirements and threats, the next step is to identify which risks can be applicable to the specific context.

The impact will be key to determine the overall risk of the system, but it can be also useful to within the security evaluation or the testing process, with the objective of prioritising the components or the verification of the threats that can have a major impact over the system. It is worth noting that ensuring that a system is 100% secure is too costly and it is actually not practically possible because other factors not included in the system can generate vulnerabilities like the human factor. We can also remark that in general there is a trade-off between security and applicability, and therefore, prioritisation of the risk or requirements should be part of the overall risk process [60].

An understanding of threats or security objectives can best be achieved by grouping them into categories. Security threats can be observed and classified in different ways by considering different criteria like source, agents, and motivations. Threat classification helps identify and organise security threats into classes to assess and evaluate their impacts, and develop strategies to prevent, or mitigate the impacts of threats on the system. Furthermore, threats classification allows evaluating and estimating threats risks [61]. It allows classifying threats into classes in order to group them by characteristics and suggest appropriate counter measures. In this phase, we also classify the applicable threats using a specific taxonomy with more general categories. Whereas the association of each specific threat with a general one makes it possible to identify sets of threats, understand the impact they can cause on the system and measure risk in a more organised way, the fact of mapping each threat to the general ones, can be used to facilitate the visualisation of the results in a multi-dimensional security label, in order to measure the risk associated to a category. The STRIDE represents a simple and a very popular threat model, that has already been applied within the automotive domain. Therefore, we propose using STRIDE to classify the specific threats into more general ones.

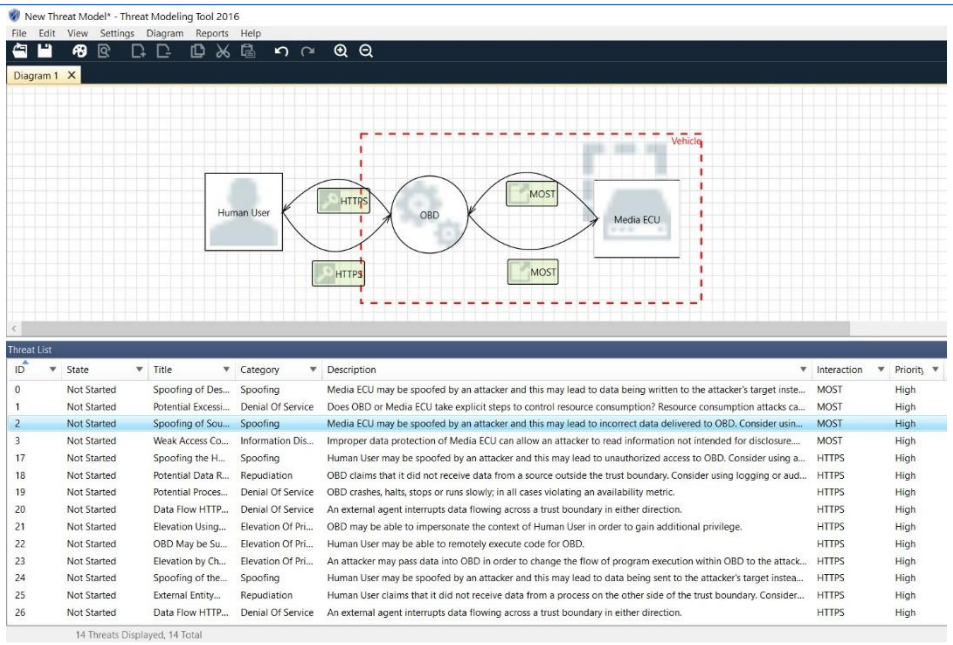


Figure 4 Use case example with Microsoft SDL tool

This phase can be partially automated using tools for threat modelling such as the Microsoft SDL tool, which has also been used within HEAVENS project. This tool is completely free and only has one dependency: Microsoft Visio. There, we can model visually each use case identified in the TOE, and the tool will generate automatically a set of possible threats that can be present in this use case. Furthermore, the tool also classifies each threat following the STRIDE model. The possibility of automating the discovery of possible threats complements in an efficient and cost effective way a manual analysis performed by a security expert.

Figure 4 shows an example of how Microsoft SDL tool can help to model a specific use case and automatically analyse possible security threats. In particular, the figure shows a use case in which the user is accessing to the on-board diagnosis using a Wi-Fi connection and the HTTPS protocol, to monitor and obtain information about the functioning of an infotainment electronic control unit, which uses the Media Oriented Systems Transport (MOST) bus (wire).

3.2.3 Combining security testing and risk assessment

The core of the methodology is the block of security risk assessment and testing, which is meant to objectively evaluate the security of the system. One of the main problems of the current assessment mechanisms, especially in the vehicle domain, is the fact that the security (and safety) assessment is carried out by a security expert, who analyses and determines the risk of a threat is present in the system. Therefore, the measurement of the risk and the verification of the safety of the vehicle, falls in an integral way in the criterion of the expert, giving rise to completely subjective evaluations. Another problem in the vehicle domain is the lack of evaluation mechanisms based on the execution of tests. While tests provide a reliable empirical mechanism through which to verify that a vulnerability is present, current approaches continue to rely solely on expert judgement. That is why the proposed methodology combines security testing and risk assessment so that the evaluation of the vehicle security can be objectively verified and the process can be repeatable, favouring transparency and comparison between vehicles.

The benefit of the interaction between risk assessment and testing is mutual, as shown in Figure 5. On the one hand, in an agile approach, the tests can be prioritised according to the impact of the vulnerability or threat that they verify, and on the other hand, the risk estimation can use the results

of the tests to refine the measurement and obtain objective metrics. Several processes are included in this integration: the first one, test implementation is intended to design and implement the tests based on the identified risk in the previous phase, selecting the most appropriate testing technique for each test; the second one, test execution prepares the environment to tests and runs the implemented tests, obtaining as a result a test report; the third one, risk estimation, takes as input the test report and computes the risk per component based on the tests related to each component; finally, the risk evaluation computes the overall risk of the system, combining the individual risks per component.

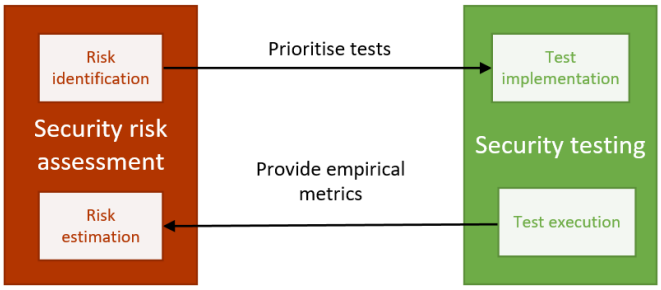


Figure 5 Mutual benefit between testing and risk assessment

3.2.4 Risk estimation

The objective of this phase is to measure the security level of the system using the information obtained from the tests. While STRIDE approach can potentially be used to identify threats in the automotive systems, DREAD model cannot be considered as a potential candidate for risk assessment in the context of the automotive industry. The ratings suggested by DREAD risk assessment model are not consistent and are highly subjective. As a result, Microsoft abandoned the model in 2008 [64]. Hence, we consider STRIDE but not DREAD for risk assessment. For measuring the risk, the usual equation combines the likelihood of exploiting the threat and the impact if this threat is exploited:

$$Risk = Likelihood * Impact$$

This formula, and specifically the two factors involved in it, has been refined, deriving in a plethora of different risk assessment schemes such as CVSS, CWSS, DREAD, etc., as reviewed in subchapter 3.3. However, whereas impact can be, in some way, estimated, likelihood is still a complex measure that requires either a historic of the threats that the system has suffered or the judgement of an expert who determines, based on several factors (e.g., equipment, necessary knowledge, exposure of the system, etc.), the likelihood that this threat is exploited. The proposed methodology will remove this issue by obtaining the likelihood directly from the security testing process and combining it with the impact. In this sense, the likelihood will take three possible values as indicated in the following Table 3:

Table 3 Test result and likelihood value

Test Result	Likelihood value
PASS	0
FAIL	1
Specific Metric	Measure scaled from 0 to 1

If the test is deterministic, in the sense that the only possible results of the test is FAIL or PASS (e.g., a sniffer can see the communications in clear), the likelihood will be 1 or 0 respectively, meaning that if ciphering is used, a sniffer entity will never be able to see transmitted data in clear, and if ciphering is not used, for sure a sniffer will see data in clear. The same happens if the test is related with checking that a specific threat from the NVD is present or not in the system. However, we can also define more elaborated tests, that instead of obtaining a boolean result, they obtain a specific measure of a metric (e.g., the percentage of ciphered data in the communications or the strength of the key lengths and algorithms used). In this case, the measure can be scaled between 0 and 1 to fit the likelihood scale. It is worth noting that both type of tests can be combined or used for refining purposes, as for example, there is no sense on measuring the percentage of ciphered data or checking the algorithm strength if no ciphering is being used.

On the other hand, the impact is obtained by the security expert. It is worth noting that each system in each context can lead to an impact of a different nature and to a different degree. A failure in a component that is directly related to the steering wheel of the car, which could affect the safety of the person, is not the same as a failure in a component of the opening of the doors, which would lead to a more monetary impact. Therefore, it is necessary to consider impact as a multidimensional measure. In particular, we consider as part of the impact the following aspects: environmental, financial, operational and privacy and legislation. Indeed, this approach has already been applied in other strategies such as HEAVENS, EVITA, MoRA or OWASP.

For example, Table 3 and the following tables shows the metrics used for safety impact. These tables are shown in this deliverable only as an example of the methodology and they are not necessarily related to the TARA methodology used in DIAS. On the other side, tampering with the automotive regulation can produce safety hazards and it is important to show a methodology, which can model safety risks and their impact. The division into different criteria (from ISO 26262-3 standard) allows a clear distinction between the metrics, ranging from no injury (0 impact) to fatal injuries and survival uncertain (1000 impact). Table 4 does the same for the financial impact, following the classification of the BSI-Standard 100-4, from no effect over the organisation to the possibility that the company disappears. In Table 5, we show the impact related with the normal operation and functioning of the system, as in FMEA. Whereas no discernible effects are evaluated with zero impact, major disruptions of the system represent a high impact (100). Finally, Table 6 summarises the criteria of HEAVENS for the privacy and legislation impact values, from no effect to privacy violations causing significant consequences for business operation, financial and for the trust and reputation of the victim. We can observe that financial and safety impact reach higher values than the other impact aspects, reflecting their importance.

Table 4 Safety impact

Safety	Criteria
0	No injury
10	Light and moderate injuries
100	Severe and life-threatening injuries (survival probable)
1000	Life-threatening injuries (survival uncertain), fatal injuries

Table 5 Financial Impact

Value	Criteria
0	No discernible effect. No appreciable consequences.
10	The financial damage remains tolerable to the organization
100	The resulting damage leads to substantial financial losses, but does not threaten the existence of the organization
1000	The financial damage threatens the existence of the organization

Table 6 Functional Impact

Value	Effect	Criteria
0	No effect	No discernible affect
1	Minor/Moderate disruption	Appearance item or audible noise
10	Moderate/Significant disruption	Degradation or loss of secondary function (vehicle still operable, but comfort or convenience functions don't work or work at a reduced level of performance). Degradation of a primary function (vehicle still operates but at a reduced level of performance)
100	Major disruption/Fails to meet safety or regulatory requirements	Loss of primary function (vehicle inoperable, but does not affect safe vehicle operation). Potential failure mode affects safe vehicle operation with some warning or non-compliance with government regulations.

Table 7 Privacy and legislation impact

Value	Criteria
0	No discernible effects in relation to violations of privacy and legislation.
1	Privacy violations of a particular stakeholder, which may not lead to abuses (e.g., impersonation of a victim to perform actions with stolen identities). Violation of legislations without appreciable consequences for business operations and finance.

10	Privacy violations of a particular stakeholder leading to abuses and media coverage. Violation of legislations with potential of consequences for business operations and finance (e.g., penalties, loss of market share, media coverage).
100	Privacy violation of multiple stakeholders leading to abuses. Such a level of privacy violation may lead to extensive media coverage as well as severe consequences in terms of loss of market share, business operations, trust, reputation and finance for OEMs and fleet owners. Violation of legislations causing significant consequences for business operations and finance (e.g., huge financial penalties, loss of market share) as well as extensive media coverage.

The resulting impact level is calculated summing all the values from the different impact aspects, and mapped to a unified quantitative value, as specified in Table 8.

Table 8 Overall impact

Sum of the impact aspects	Impact level	Impact Value
0	No impact	0
1-6	Low	1
7-13	Low	2
14-19	Low	3
20-45	Medium	4
46-73	Medium	5
74-99	Medium	6
100-299	High	7
299-699	High	8
699-999	High	9
>=1000	Critical	10

The same table also shows the equivalent qualitative impact level in the HEAVENS approach. Combining the impact level with the likelihood obtained from testing, we can obtain for each threat, the risk associated to it. Using the arithmetic mean, we can calculate the total risk associated to a component of the system if we found n threats as shown in the equation below.

$$R_{Component} = \frac{\sum_{i=1}^{n} Likelihood_{Threat} * Impact_{Threat}}{n}$$

The risk will then range between 0 (no risk) and 10 (critical risk). Then, the next step is to combine the individual risks of the components to obtain the overall risk of the system. Different approaches can be followed to perform this step. The easiest one is to combine again the risk per component using

the arithmetic mean. However, it should be noted that not all components are equally important within a system. While a security failure in the audio system can lead to an interruption of service, a security failure in the brakes or the steering wheel can affect the entire system, and can even lead to the loss of human life. Furthermore, certain components can be key to the operation of the system, such as the CAN bus, so a failure in it would affect a wide number of components of the system. Therefore, including in the system risk measurement a weight that indicates the impact that a component may have on the system is key to reflect the security dependencies. The impact of the component over the system, which should range between 0 and 10, is intended to be specified during the risk identification phase, when the use cases and dependencies are analysed. This value can be specified directly by the manufacturer or the security expert or obtained by an analyst using the use cases diagrams or additional techniques such as attack graphs. We assume here that this value has been already obtained. Therefore, the overall risk of a system composed by m components is obtained in the following way:

$$R_{system} = \sum_{i=1}^{m} R_{component} * Impact_{component}$$

As before, the overall risk will range between 0 (no risk) and 10 (critical risk). We can also reflect the security level (SL) by inverting the risk value, and, in this case, the SL will range between 0 (bad security level) and 10 (very good security):

$$SL_{system} = 10 - R_{system}$$

3.2.5 Treatment

According to ISO 3100, "Risk treatment is a risk modification process. It involves selecting and implementing one or more treatment options. Once a treatment has been implemented, it becomes a control or it modifies existing controls". In general, the results of the security evaluation are used only to validate or certify the security of the system, missing a very valuable information that reports the security flaws that our system has and how they could be avoided during the system's operation phase. In this sense, we propose as a way to address risk treatment, the creation of a behavioural profile with some recommendations (polices) to take into account during the operation phase. This profile is intended to reduce the attack surface to the allowed behaviours, and it could be also used to monitor suspicious behaviours during the operation phase. This activity inter-operates and makes use of the results of the previous evaluation, as the behavioural profile is intended to be generated from the security results containing both security recommendations from the manufacturer and from the security evaluation to perform a secure deployment. In particular, we propose to base the behavioural profile on a standardized format such as the Manufacturer Usage Description (MUD), due to its flexibility and scalability.

MUD is an Internet Engineering Task Force (IETF) standard aimed to define the intended behaviour of the device through Access Control Lists (ACLs), in order to restrict the communication to/from a certain device. MUD defines an architecture for obtaining MUD files wherein those policies are specified by using the Yet Another Next Generation (YANG) and JavaScript Object Notation (JSON) standards. While MUD was recently standardised (March 2019), it has received a strong interest from the research community and standardisation entities worldwide. The USA National Institute of Standards and Technology (NIST) has also recommended MUD files to complement security credentials in order to reduce the attack surface. Although this is outside the scope of the methodology, the reader can refer to [65] to get some insights on how to generate, obtain and enforce the MUD during the operation phase.

3.2.6 Monitoring and communication

During the operation phase, the system and its components are providing the functionality for which they were manufactured. In this phase, the system should be monitored, since new security vulnerabilities can be discovered or a new patch/update can be installed, and consequently, the system's security level can be modified. Both the changes produced by an updating process, and the modifications produced by an unexpected event (e.g., the discovery of a new vulnerability) led to a new security level, so a continuous reassessment should be done, starting a re-evaluation process if needed. In this context, the deployment of security solutions to control the behaviour of the system components and networks is essential to improve the reaction time by the infrastructure managers and the fast mitigation of vulnerabilities.

The realisation of an effective detection of security attacks in a specific system or network requires identifying the expected behaviour of each device or component composing such environment [66][67]. Indeed, most of existing approaches based on machine learning techniques to improve security [49] require the proper definition of devices' intended operation and behaviour to train the corresponding model. The concept is that events or communications, which are not part of the device's normal behaviour, can be considered as a potential threat or attack. From another point of view, a legitimate behaviour may be imposed on devices. For example, rules can be defined and applied to determine how a device is deployed or connected to a network. For that purpose, specific network components may require adapting their operation to enforce restrictions associated with the intended operation of a new device.

Linking with the previous chapter, and taking advantage of the evaluation results, the behaviour profile represents a very powerful tool to monitor that the behaviour of the system and its different components in the execution phase is as expected. If a deviation from the expected behaviour is detected that could lead to an imminent attack, the appropriate mitigation could be applied to prevent said attack. Monitoring, IDS and mitigation techniques are crucial in a domain as sensitive as that of smart vehicles, in which a security breach can end with a loss of human life. In this context, the exchange of security information between all stakeholders becomes even more important. Knowing as soon as possible the existence of a vulnerability or installing a patch that mitigates it is crucial to maintain the security of the system throughout its life cycle.

3.3 Risk Assessment approaches

Risk assessment is defined in CNSSI-4009 as "the process of identifying, prioritizing, and estimating risks. This includes determining the extent to which adverse circumstances or events could impact an enterprise. Uses the results of threat and vulnerability assessments to identify risk to organizational operations and evaluates those risks in terms of likelihood of occurrence and impacts if they occur. In this sense, this document considers a risk assessment methodology as a process to determine the risk of a vulnerability, weakness or threat.

3.3.1 CWSS

The Common Weakness Scoring System (CWSS) [16] provides a mechanism for prioritizing software weaknesses and assign a numerical risk to them. To do so, CWSS combines three groups of metrics that are used to calculate the risk: Base Finding, Attack Surface, and Environmental.

- The Base Finding: is focused on the inherent risk of the weakness, the confidence in the accuracy of the finding, and strength of controls.
- The Attack Surface: includes factors representing the barriers that an attacker must exceed to exploit the weakness.

- The Environmental: groups characteristics of the weakness that are specific to a particular environment or operational context.

Each factor in the metric groups is assigned a value, which is converted to its associate weight. The metrics of each group is calculated and combined with the other groups (multiplication) in order to obtain a complete risk measure, which ranges between 0 and 100. The Base Finding subscore is between 0 and 100, whereas the other ones can range between 0 and 1. The main advantage of CWSS is that it allows unknown values when the information is incomplete, so it can be applied earlier in the process, before any vulnerability has been proven but it should be noted that, in spite of exhaustively defining these metrics, some of them are difficult to be quantified, as highlighted by [17].

If the set of values proposed for the TI metric, is not precise enough, CWSS users can use their own quantified methods to derive a subscore. One of the methods uses the Common Weakness Risk Analysis Framework (CWRAF) [18] to define a vignette and a Technical Impact Scorecard. Here, vignette-specific Importance ratings are used to calculate the Impact weight. CWRAF and CWSS allow users to rank classes of weaknesses independent of any particular software package, in order to prioritize them relative to each other (e.g., "buffer overflows are higher priority than memory leaks"). This approach, sometimes referred to as a "Top-N list," is used by the CWE/SANS Top 25 and OWASP Top Ten.

CWSS is recommended by the ITU-T1 and it is used in several databases such as the CWE2 or the OWASP Top Ten3.

3.3.2 CVSS

The Common Vulnerability Scoring System (CVSS) [19] is a widely adopted methodology which captures the main characteristics of a vulnerability and provides to users a common and standardized scoring system within different cyber and cyber-physical platforms.

Similar to CWSS, since it also consists of three metric groups:

- The Base metric group: represents the intrinsic characteristics of a vulnerability that are constant over time and across user environments; the exploitability metrics reflect the ease and technical means by which the vulnerability can be exploited; the impact metrics measure how a vulnerability, if exploited, will affect the vulnerable component. Based on the Base Metric Group, CVSS produces a numerical severity score ranging from 0.0 to 10.0, which can be modified by scoring the optional Temporal and Environmental metrics (they include a metric value that has no effect on the score).
- The Temporal metric group: reflects the characteristics of a vulnerability that may change over time but not across user environments.
- The Environmental metric group: represents the characteristics of a vulnerability that are relevant and unique to a particular user’s environment.

Although CVSS is similar to CWSS, some metrics like likelihood have been removed, leading to simpler to calculate metrics. There are also some differences between usage scenarios of CVSS and CWSS. CVSS assumes that a vulnerability has already been discovered and verified, while CWSS can be applied earlier in the process, hence before any vulnerabilities have been proven. In addition, CVSS scoring does not account for incomplete information, while CWSS scoring has built-in support for also incomplete information. Compared to other risk assessment rating processes like DREAD, CVSS focuses mainly on consistently having accurate measurements of a system’s vulnerabilities and is more comprehensive with regards to its contributing factors to determine a systems risk rating.

CVSS has been widely adopted, especially the use of base scores from the Base metric group and it represents a widely established approach; for example, it is used in the CVE and in the National Vulnerability Database (NVD) created by the NIST.

3.3.3 OCTAVE

The Operationally Critical Threat, Asset, and Vulnerability Evaluation (OCTAVE) [20] was proposed by the Carnegie Mellon University's Software Engineering Institute (SEI) in collaboration with the Computer Emergency Response Team (CERT) in USA. The methodology covers different aspects for risk management, including the identification of relevant assets for a certain organization, and the vulnerabilities and threats associated to such assets, in order to define a strategy accordingly. It should be noted that OCTAVE focuses on operational risk and security practices, not on technological aspects related to the risk to be assessed. Therefore, it is not easily applicable to most vehicular systems, considering also the high amount of documentation, training and practices that are necessary to apply it.

The OCTAVE framework defines eight processes. However, before performing such processes an exploratory phase or Phase Zero is used to come up with the criteria to be used during the application of the methodology. While it defines a set of metrics, the mapping with impact intervals (low, medium and high) is open to subjective interpretation [20], so that the comparison between different OCTAVE-evaluated components is more difficult. In addition, the methodology is based on a complex documentation, which has motivated the development of a more lightweight alternative called OCTAVE-S [21].

3.3.4 DREAD

The DREAD scheme [21],[22] is used to compute a risk value associated to a certain threat or vulnerability based on the use of five categories: Damage potential, Reproducibility, Exploitability, Affected users and Discoverability. It was used at Microsoft and currently it is used by OpenStack [23]. The context can be considered when assigning a value to each category.

When a given threat is assessed using DREAD, each category is given a rating. The DREAD algorithm is then used to compute a risk value, which is an average of all five categories.

The calculation always produces a number between 0 and 10; the higher the number, the more serious the risk. The sum of all ratings for a given issue can be used to prioritize among different issues.

However, according to [24], it is not clear how the scale should be considered; indeed, it depends on the users responsible for performing the modelling of threats. DREAD requires scoring each of the five categories on a scale from 0 to 10, which leads to discussions on the fine differences between consecutive numbers, e.g., five and six. This problem is exacerbated when the assessment is performed by different users that must agree on these aspects. A potential solution to this issue, as remarked in [25], is the use of scores of High, Medium, or Low, that are easy to agree, instead of using the eleven-valued scale by Microsoft.

3.3.5 OWASP Risk Rating Methodology

The OWASP Risk Rating Methodology [26] is part of the OWASP project, which provides a basis for testing web application technical security controls. The risk rating methodology estimates the risk in terms of likelihood and impact considering 4 main factors (Threat Agent, Vulnerability Factors, Technical Impact and Business Impact). To do so, OWASP follows several steps. The first one consists on identifying a risk to be rated, analysing and gathering information about it. The second step analyses factors for estimating likelihood. It is not necessary to be over-precise in this estimate. Generally, identifying whether the likelihood is low, medium, or high is sufficient. There are a number

of factors that can help determine the likelihood, such as the ease of discovery and exploit or the skills of the attacker. The third step is about identifying factors for estimating Impact, divided in technical impact on the application, the data it uses, and the functions it provides and in business impact on the business and company operating the application. In this sense, the context factor can be considered through this metric. The fourth step determines the risk severity. The likelihood and impact estimate are put together to calculate an overall severity for this risk, obtaining none, low, medium, high or critical. Finally, it is decided what to Fix. It is also possible to customize the Risk Rating Model, for example adding factors, customizing options or weighting the factors. The limitation of OWASP is that it is only focused on web applications. As in the other schemes, the scale (low, medium and high) is only based on the consensus of the users, so that a certain degree of subjectivity is associated to this process. Furthermore, OWASP has a very low connection with the assets involved, which in web applications make sense because most of the information about assets involved is common knowledge and there is no need for a special equipment to communicate with the application. However, the case is different in automotive systems, where not all knowledge about assets is public and custom tools are needed to access interfaces [27].

3.3.6 Veracode Rating System

The Veracode Security Quality Score [28], is a scoring system based on CWE dictionary of security flaws, used to map the flaws found in its static and dynamic scans, and on CVSS for the calculation of severity based on the potential Confidentiality, Integrity and Availability impact of a flaw CWE if exploited. Each severity level reflects the business impact if a security breach occurs in these three security aspects. It is part of the Veracode Platform, which uses static and Dynamic analysis (for web applications) to inspect executables and identify security flaws in applications.

Veracode assigns a severity level to each flaw type:

- Confidentiality Impact: measures the impact on confidentiality if on a system vulnerability is exploited. At the weakness level, the Confidentiality is measured at three levels of impact: None, Partial and Complete, in according CVSS;
- Integrity Impact: measures the potential impact on integrity of the application. Integrity measures are needed to protect data from unauthorized changes. In fact, when the integrity of a system is solid, it is protected from unauthorized modifications of its contents.
- Availability Impact: measures the potential impact on availability if an attack is successful on the vulnerability. The Availability means to the accessibility of information resources. Typically, in this domain, the vulnerabilities are the Denial of Service. For example: Attacks that compromise authentication and authorization for application access, application memory, and administrative privileges.

The overall Security Quality Score, based on its associated CWE entry, is computed by aggregating impact levels of all weaknesses within an application. It enumerates the security weaknesses and their impact levels within the application code, but it does not predict the potential for vulnerability. It is a single score ranging from 0 to 100, where 0 is the insecure application and 100 is an application where the flaws have been discovered. The score calculation includes non-linear factors so that, for instance, a single Severity 5 flaw is weighted more heavily than five Severity 1 flaws, and so that each additional flaw at a given severity contributes progressively less to the score. It should be noted that this score is then weighted according to security experts, so a certain degree of subjectivity is included in the process.

3.3.7 Cenzic Harm

The Cenzic Hailstorm Application Risk Metric (HARM) [29] is a quantitative metric for the risk is associated with a web application. It is split into 4 impact areas: Browser, Session, Application, and Infrastructure (server environment). It also takes into account two additional factors, a complexity factor and the precision associated with detection of a given vulnerability and a modifier called weight, which users can use to modify the obtained risk.

To determine the application risk level (impact value) for a vulnerability, HARM uses security values with five degrees of risk such as confidentiality or access. The vulnerability risk is the sum of the risk score from each of the four impact areas, which can be modified by the weights from other metrics (i.e., attack complexity, detection precision, asset value).

Finally, the HARM rating is calculated by multiplying all of the identified vulnerabilities (that can include different components and layers) within an application by the level of importance managers give to that application, so it gives the possibility of indirectly considering the context changing the weights.

However, this method does not account for the relationship of vulnerability properties, which are also important in the evaluation of the distribution of exploitation, and it is focused only on web applications.

3.3.8 Common Criteria

The most well-known cybersecurity certification standard is the Common Criteria (CC) [30], in which the security functional and assurance requirements are specified through Protection Profiles (PPs) for a Target of Evaluation (TOE), which is a set of software, firmware and/or hardware. These requirements are defined in the Security Target (ST) description.

CC permits comparability between the results of independent cybersecurity evaluations providing a common set of requirements for the security functionality of IT products applied to these IT products during a security evaluation. This is obtained through the Collaborative Protection Profiles (cPP), which become available for use under the terms of the Common Criteria Recognition Arrangement (CCRA) [31]. Evaluations conducted against cPPs on this list are mutually recognized according to the terms of the CCRA. The evaluation process establishes a level of confidence that the security functionality of these IT products and the assurance measures applied to these IT products meets these requirements. The evaluation results may help consumers to determine whether these IT products fulfils their security needs. For this purpose, it uses Evaluation Assurance Levels (EAL) to describe numerically the depth and rigor of an evaluation. Basically, CC assurance is achieved by carrying out analysis and checking of processes and procedures, guidance documents, TOE design, functional tests, independent functional testing, vulnerabilities and penetration testing [32][33].

Although CC is the main standard, the community has identified several limitations that are being considered [32][33]. Examples of them are the time and effort required to document the evaluation and to gather evidence, in particular at high EALs, or the management of changes in the certified product. During the manufacturing process, this could involve market delays and therefore, considerable financial loss. It is worth noting that CC evaluation is focused on a specific version of the TOE, including the configurations. This means that any change over in the TOE (e.g., a new vulnerability) could invalidate the result of the certification, something that is quite critical in frequent updated products. In addition, recertification is not mandatory in CC, and the responsibility of informing about security changes lies with the owner of the certificate.

Regarding the smart vehicles domain, the Car-to-car communication consortium led an initiative to define a PP for vehicle communication devices [34]. The PP may not address all the categories of good practices. However, the integration of the CC scheme ensures the security assessment by skilled third-party laboratories, supervised by national cybersecurity agencies, following a standard process.

The following Tables provides a comparison among general risk assessment schemes:

Table 9 Comparison among general risk assessment schemes

Scheme	Advantages/Features	Disadvantages
CWSS	• Standard and widely used in several lists and databases such as OWASP or CWE • Applicable early in the process • Easy to apply • Possibility of more precise metrics with vignettes (CWRAF)	• Qualitative metrics • Metrics difficult to calculate • Same weight to metrics easy and hard to calculate
CVSS	• Standard and widely used in several lists and databases • Simple metrics • Easy to apply	• Qualitative metrics • Same weight to metrics easy and hard to calculate
OCTAVE	• Covers different aspects of risk management • Lightweight version OCTAVE-S	• Not standard and not widely used • Complex documentation to draft • Qualitative metrics • Certain degree of subjectivity in the qualitative intervals
DREAD	• Easy to apply • Linked with the STRIDE taxonomy	• Not standard and not widely used • Qualitative metrics • Certain degree of subjectivity in the qualitative intervals
OWASP	• Easy to apply • Useful for web apps	• Not standard • Qualitative metrics • Domain-specific (web apps) • Certain degree of subjectivity in the qualitative intervals
VERACODE	Easy to apply Includes non linear factor for severity Calculation based on the CWE entry	• Not widely used and not standard • Certain degree of subjectivity in the qualitative intervals • Qualitative metrics
CENZIC	Usage of personalised weights to aggregate risks within an application.	• Not standard • Domain-specific (web apps) • Qualitative metrics
COMMON CRITERIA	Standard wide used (CCRA) Usage of collaborative protection profiles to specify the security objectives Usage of Evaluation assurance levels to facilitate comparison	• Difficult to compare due to technical documents • Costly and time consuming • Qualitative metrics

3.4 Specific Risk Assessment activities in the automotive sector

The aim of this chapter is to describe the most common risk assessment activities in the automotive sector. While a risk assessment methodology was already described in DIAS deliverable D1.1, this deliverable D4.4 gives an overview of the validation and verification approaches for continuous testing, where risk assessment is an important part because it drives the definition of requirements for V&V and continuous testing, which is then used to re-evaluate the risks again.

3.4.1 Modular Risk assessment

The basis of Modular Risk Assessment (MoRA) [35] is the determination of individual protection objectives, that is, those parts of a system that are particularly worth protecting. This determination requires close coordination with the respective company to better apply the risk analysis to the specific system. The methodology has been also applied in the context of smart vehicles, as it provides a high flexibility, also supporting hierarchical decomposition of the target of evaluation [35].

In particular, MoRA comprises four main activities: 1) Document TOE, 2) Determine Protection Needs, 3) Analyze Threats, and 4) Analyze Risks. Potential Threats as well as the related controls and countermeasures are analyzed and identified by using catalogs of known threats or vulnerabilities to evaluate if there is a matching. Then, security solutions like controls and countermeasures can be selected accordingly to the identified threats. In MoRA, threats are connected with components or data flows and evaluated based on risk factors like time, access, knowledge and equipment. These activities are supported by the SecurityAnalyst tool, a proprietary tool, which was developed by Fraunhofer together with Itemis AG. The objective is to evaluate the security of existing systems using a risk model in a uniform, comparable and understandable way and to propose measures to minimize the risk. The tool provides a set of guidance artefacts such as Assessment Model or Risk Assessment Template that can be used to specify a particular assessment model. Such tool provides results to indicate the estimated likelihood of risks. The methodology itself just gives the indication on the steps that should be performed but it does not give indications on what tools or approaches can be useful to instantiate the methodology itself beyond proprietary implementations. Moreover, the tool does not provide a high automation of the processes, as the user is in charge of modelling, specifying the risks and selecting the mitigation. Finally, the methodology is based on the expert's opinion to quantify the risk instead of relying on security testing to validate the compliance of the security objectives.

3.4.2 E-safety Vehicle InTrusion protected Applications (EVITA)

The E-safety Vehicle InTrusion protected Applications (EVITA) [36] risk is evaluated based on the severity and attack probability (attack potential). The attack severity of an attack includes four different aspects that may be associated with harm to the stakeholders (operational, safety, privacy, and financial aspects) as a 4-component vector with a range of qualitative levels. The severity of an attack is assessed using attack trees, by considering the potential implications of the attack objectives for the stakeholders. The attack potential is a measure of the minimum effort to be expended in an attack to be successful, and it includes four different factors: elapsed time to identify the vulnerability and develop the attack, specialist expertise required, knowledge of the system under investigation, window of opportunity (e.g., if access should be continuous, if online preparation is needed...) and IT hardware/software or other equipment needed. The probability of a successful attack is also assessed using attack trees, by identifying combinations of possible attacks on the system assets that could contribute to an attack method.

Finally, the risk level is determined from the severity and the combined attack probability associated with a particular attack method. This is achieved by mapping the severity and attack probability to the

risk using a "risk graph" approach. Towards this end, the probability and severity combinations are mapped to a series of risk levels ranging from 0 (lowest) to 6 (highest).

Although EVITA is very similar to OWASP's methodology, EVITA is more tied to the assets involved though the threat agent factors (Knowledge of the target, equipment required), making it more suitable for the automotive industry.

3.4.3 Modified CVSS

The author of the thesis in [27] argues that the CVSS methodology gives equal weight to all parameters, assuming that we know all the parameters with a high confidence degree. If we decompose the CVSS metrics in CVSS severity, exposure, and exploitability, all of them have the same priority. However, while the severity of a component is easy to determine, exposure and exploitability are highly subjective. In addition, exposure and exploitability are interconnected. A component's exposure dictates how exploitable it is. Hence, the author proposes a CVSS modified formula that gives a higher priority to parameters that can be deduced accurately and less priority to parameters that cannot:

$$Risk = \frac{Severity \times (Exposure + Exploitability)}{10}$$

3.4.4 Modified OWASP

In the same thesis [27] mentioned in the previous subsection, the author also proposed an adapted version of OWASP to smart vehicles. Inside the Threat Agent factor, which is part of the likelihood component, OWASP considers the Skill, Motive and Opportunity of the attacker. However, the author argues that Skill and Motive are related remain constant depending upon the attacker, whereas Opportunity may depend upon the component and its ease of access. Therefore, in the adaptation, skill and motive were converted into a generic item, instead of having a skill and motive component for each asset.

3.4.5 HEAVENS

HEAling Vulnerabilities to ENhance Software Security and Safety (HEAVENS) [37] defines a framework for identifying security requirements in the context of smart vehicles, representing an improvement of the EVITA methodology. Towards this end, the HEAVENS methodology identifies assets and threats associated with the assets and then, threats are mapped with the security attributes, deriving a security level for each asset-threat pair. In particular, HEAVENS considers the following security requirements: confidentiality, integrity, availability, authenticity, authorization, non-repudiation, privacy, and freshness.

HEAVENS analyses threats based on Microsoft's STRIDE approach and ranks the threats based on a risk assessment. The security level is calculated using two metrics: threat level and impact level. Similar to the Method of EVITA, the threat levels, which reflect the likelihood of the threat, are computed based on the same parameters used in Common Criteria. it uses a scale of 4 to 0, with 4 being very high and 0 being low likelihood. The scale for impact level ranges from 4 to 0 as well, with 4 being critical and 0 being Quality Management.

The impact of the threats is quantified by considering the expected loss of the objectives, which are safety, finance, operation and privacy. The STRIDE model is used for threat analysis.

HEAVENS improve EVITA in several ways. For example, EVITA does not distinguish among various access types (e.g., physical, logical) while rating opportunity and although the impact level is aligned with ISO 26262, it does not provide a suitable guideline to estimate the impact and it does not take other legislation aspects into account for risk rating. In this sense, HEAVENS provides estimation of

impact level parameters (safety, operational, financial, privacy and legislation) based on industry standards. For example, the safety parameter is aligned with the ISO 26262, financial parameter is based on the BSI-Standard, and operational parameter is based on the Failure Mode and Effect Analysis (FMEA) proposed by the Automotive Industry Action Group (AIAG). Last but not the least, the EVITA approach is attack-centric and focuses on identifying all possible attacks against a TOE although attacks are dynamic and practically, possible attacks and attack techniques are virtually endless.

However, HEAVENS security model also has some limitations. It does not suggest countermeasures or security mechanisms to assist in fulfilling the derived security requirements, and empirical security measurement (e.g., testing) is neither considered.

3.4.6 Threat, Vulnerability, and Risk Analysis (TVRA)

Threat, Vulnerability, and Risk Analysis (TVRA) [38] is an assessment method developed by the European Telecommunications Standards Institute (ETSI) developed for data and telecommunication networks.

The TVRA method can briefly be summarized as follows; First, The TOE, the associated assets (physical, human or logical) and the goals of the evaluation are identified. Security objectives are then identified and classified based on the five security attributes: confidentiality, integrity, availability, authenticity, and accountability. From them, we can derive the functional security requirements, which are more detailed requirements than the security objectives, e.g., passwords should be used for authentication. An inventory of assets is done and possible vulnerabilities are identified and classified along with corresponding threats and undesirable results. These threats are classified according to the following four categories: interception, manipulation, denial of service and repudiation. The risk is calculated based on the likelihood of these threats and their undesirable results. Finally, a set of countermeasures are derived and cost- benefit analysis is carried out to select the most appropriate countermeasures to reduce the risk of the identified threats. These results are then used to design security services. However, the standard only gives the steps to be performed, not how to perform them. Moreover, although the method has already been applied in the vehicular context, the process is quite large and complex [39].

The TVRA is a similar concept to the Threat Analysis and Risk Assessment (TARA) from ISO/SAE 21434 [124] and J3061 [101].

3.4.7 Other approaches

Beyond previous approaches, other works propose the development of new risk assessment methodologies that can be potentially considered for smart vehicles. In this subsection we mention some of them, but a more extended review and comparison of current risk assessment methods for smart vehicles can be found in [39] and [40].

Based on CVSS, authors in [41] propose a method based on Bayesian net- works and machine learning to obtain the risk associated to a vulnerability. Although the authors apply the method to smart vehicles, it is general and can be applied to any other scenario. In [42], the authors propose an approach for aligning automated vehicles safety and security at early development phases by synchronizing safety and security life-cycles based on SAE J3061, SAE J3016 and ISO 26262 standards. Towards this end, they use Failure Attack and Countermeasure (FACT) graphs to list safety failures, security attacks and the associated countermeasures. SAHARA is a security-aware Hazard Analysis and Risk Assessment method (HARA) that encompasses threats from STRIDE model. SAHARA proposes a security level determination method, and uses it in combination with Automotive Safety Integrity Levels (ASILs) to assess the possible threats. The SAHARA quantification scheme is less complex and requires less analysis effort and fewer details of the analyzed system than other proposed approaches

[39]. Attack trees are also used in [43] to logically model hacking attacks against vehicles and to evaluate the security against threats and vulnerabilities, and in [44], in which a systematic threat analysis and risk assessment framework, SARA, is proposed. Moreover, SARA involves the attacker in the attack tree, and it is applied to two particular use cases related with smart vehicles. The Failure Mode Vulnerabilities and Effect Analysis (FMVEA) [45] method analyses safety and security cause-effect. This work categorizes threats via quantification of threat agents (respectively attacker), threat modes (via STRIDE model), threat effects and attack probabilities. Combined Harm Assessment of Safety and Security for Information Systems (CHASSIS) method [46] also combines safety and security methods for a combined safety and security assessments approach. The approach relies on modelling misuse cases and misuse sequence diagrams within a UML behaviour diagram, to define whether there are features that are mutually dependent or independent of each other.

3.5 Threat taxonomies

The first step in risk assessment is to understand and identify the threats (if we think in negative) or the security objectives (if, on the contrary we think in positive). In fact, threats represent events that cause a system to respond in an unexpected or damaging way, and the security objectives represent the countermeasures established to avoid these threats. An understanding of threats or security objectives can best be achieved by grouping them into categories [47].

Security threats can be observed and classified in different ways by considering different criteria like source, agents, and motivations. Threat classification helps identify and organize security threats into classes to assess and evaluate their impacts, and develop strategies to prevent, or mitigate the impacts of threats on the system.

The CIA triad (confidentiality, integrity, and availability) represents the basic classification of security objectives, possibly mentioned first by the NIST in 1977 [48]. However, debate still questions whether it is sufficient to handle rapidly changing technology and business requirements, with recommendations to consider expanding on the intersections between availability and confidentiality, as well as the relationship between security and privacy. In this sense, many experts usually add other principles [49] such as non-repudiation, privacy, authenticity and trustworthiness, accountability and auditability. Three Tenets of Cybersecurity [50] have been identified by DoD Software Protection Initiative starting from 2009, indicating System Susceptibility, Access to the Flaw, and Capability to Exploit the Flaw while Open Group proposed in 2011 the operational definition of security objectives in its security management standard O-ISM [51], assessing information control and availability, data quality and compliance.

Focusing on threats, more than in security objectives, the Network Security Threat Classification proposed in [52] considers four different types: Unstructured Threats, caused by inexperienced individuals that uses hacking tools such as shell scripts and password crackers; Structured Threats, caused by highly motivated and competent hackers. In this case, the techniques and tools used are more sophisticated and complex; External Threats, caused by unauthorized individuals or organizations working outside of a company; Internal Threats, caused by an authorized individual (e.g., an account on a server or physical access to the network); This classification is highly intuitive, allowing to identify and classify network threats and vulnerabilities. However, some of the categories overlap; for example, a threat caused by an inexperienced user can be both external or internal. Moreover, this classification did not cover all threats, as they just present network security threats.

The ISO 7498-27 standard has listed five major security threats and services as a reference model: Destruction of information and/or other resources, Corruption or modification of information, Theft, removal or loss of information and/or other resources, Disclosure of information and Interruption of

services. This classification is mutually exclusive, but it is highly focused on information threats, and it does not cover all threats consequences.

STRIDE is a model of threats developed by Praerit Garg and Loren Kohn-Felder at Microsoft for identifying computer security threats. It provides a mnemonic for security threats in six categories (Spoofing, Tampering, Repudiation, Information disclosure, Denial of Service and Elevation of Privilege), that can be understood in a positive way with its associated security property. STRIDE model is a simple and a very popular threat model, and it highlights many top threats. Besides, this classification allows organising a security strategy to reduce risks. Although it is neither exhaustive, STRIDE has been used in the smart vehicles' domain within the HEAVENS project, and the threats considered in EVITA can be also mapped to the ones of STRIDE [27].

The Web Application Security Consortium Threat Classification [53] classifies threats in six categories: Authentication, Authorisation, Client-Side At- tacks, Command Execution, Information Disclosure and Logical Attacks. This model is quite flexible and shows the direct impact on security requirements if a threat happens, which help to make appropriate countermeasures. However, the list is not exhaustive, as it is focused on web site threats.

3.6 Challenges in the implementation and execution of risk assessment processes

This chapter analyses and describes the main challenges of cybersecurity risk assessment, with the focus on cybersecurity evaluation aspects and the application on the smart vehicles' context. This analysis is based on the inputs of several cybersecurity organizations, such as ENISA1 or ECSO, as well as in the lacks observed in the current state of the art.

3.6.1 Security Composition

A smart vehicle is composed of several components performing different functionality, and moreover, the heavily-tiered ecosystem of car manufacturing also leads to security integration issues. The wide heterogeneity of devices and technologies (e.g., machine learning, artificial intelligence, cloud computing, net- works, sensors) that can be part of the vehicle ecosystem derives on a heterogeneous environment that hardens the security evaluation. Therefore, it is crucial to detect cascade effects and assess the real security level of the system. It means that the risk assessment of a certain system will depend of the security level associated to each part of the system. For this aspect, the main challenge is related to the way in which each risk value could be aggregated to provide a reliable value for the whole system [54]. While most of the risk assessment schemes already consider the use of weights associated to different metrics to assess a certain product, these aggregation aspects are usually ignored. There is the need for more sophisticated security evaluation mechanisms that can capture components interdependence and cascade-effects among all the involved components. These mechanisms will help capture how interdependencies operate and will heighten impacts in order to develop procedures and policies to improve recovery.

3.6.2 Time and economic resources

The existing approaches for cybersecurity assessment (e.g., CC, OCTAVE, EVITA) are often time consuming and complex, requiring formal documentation and processes, which could imply that such process can impact the launch of a new product [55],[56]. From the certification point of view, a costly process could imply that the manufacturer cannot afford the costs related to the cybersecurity certification process. On the one hand, the cybersecurity certification could require monetary costs related to the payment to a certification body issuing the certificate, or the testing laboratory being involved in the process. Furthermore, the process can involve qualified personnel to implement the

measures required to obtain the certification. This issue is even more relevant in case of update, which may require a re-evaluation and re-certification.

Based on this, the cybersecurity evaluation process should be cost-effective and lightweight to foster its adoption, facing the trade-off between the evaluation assurance level (EAL) and the costs for the companies (time and money), specially for SMEs and startups. It could also help to deal with security changes, by providing a faster and affordable re-evaluation process.

3.6.3 Objectivity and reproducibility in the risk assessment process

Common risk assessment approaches are based on the use of different security metrics, which are employed to provide a more reliable assessment of a product's security level [57]. However, some of these metrics, such as likelihood, are difficult to be measured as it requires historical data or the opinion of an expert. As a consequence, the assessment process involves a certain degree of subjectivity depending on the security expert who is analysing the system. Indeed, some risk assessment schemes (e.g., CWSS) decided to ignore some of these metrics. This fact makes it difficult to compare the security acquired between different systems, since although the same scheme is used, different systems may have evaluated very differently. In this sense, there is a need to support the security evaluation of the system with objective metrics derived from empirical observations.

3.6.4 Definition of the operational context

The context where the system will be eventually deployed should be taken in consideration in the risk assessment process. As a main aspect of the cybersecurity evaluation, the context should embrace the purpose of the system, as well as the sensitivity of the data that will be managed. It should be noted that in the smart vehicles context, the target system is easily accessible for hackers as vehicles are a mass-market product, and the impact over the system could be fatal, implying consequences on the user life and on other vehicles. On the other side, the precise definition of the operational context is not easy to achieve in the automotive sector also because there are some externalities (e.g., the future automotive applications and/or the trust of the workshop), which are difficult to control by the owner of the vehicle or the manufacturer of the vehicle.

3.7 Relationship between the landscape analysis of the methodologies for risk assessment and threat modelling and the actual methodology used in the DIAS project (TARA)

The DIAS project proposed the use of Threat Analysis and Risk Assessment (TARA) as recommended in SAE J3061 [101], which was presented and discussed in Section 4.3.3. and by the ISO 21434-2021 [124] standard. The TARA goal is to identify potential cybersecurity threats of the system under test and the subsequent assessment of risks associated with the identified threats.

More specifically, it is defined in [101] as "an analysis technique that is applied in the concept phase to help identify potential threats to a feature and to assess the risk associated with the identified threats" and it is defined to be composed by three steps:

1. Threat Identification
2. Risk Assessment (includes classification of the risk associated with a particular threat)
3. Risk Analysis, which ranks threats according to their risk level.

As discussed in [136], before the application of TARA, some pre-analysis steps should be performed, including (a) the cybersecurity item definition, (b) the critical function block analysis, which describes the primary system functions, (c) the analysis of the technology stack in use, detailing the hardware and software building blocks for implementing specific cybersecurity mechanisms, and (d) threat

modelling to represent the adversarial cyber threats inherent to every cyber(-physical) system. In particular, the threat modelling can be based on various methodologies described before.

TARA has many similarities and it is related to many of the risk assessment and threat identification methodologies described in the previous subsections. In particular, it is quite similar to the TVRA described in section 3.4.6. This is also another reason why section 3.4 has provided an overview of the different methodologies and approach for risk assessment: because different methodologies can complement and reinforce each other as also discussed in the following paragraphs. In fact, SAE J3061 highlights that “It is left to an organization to determine which TARA method is appropriate for their purposes, and to determine what an acceptable level of risk means ...'.

The authors of [139] were some of the first authors to address the need to specify more in detail the implementation of the specific steps of TARA and they performed a similar analysis performed in this deliverable in section 3.4. The authors of [139] suggested the use of methodologies described before in this deliverable including EVITA, TVRA, OCTAVE and HEAVENS. In particular, the authors of [139] highlighted that EVITA is particularly suited for the automotive technologies. In [139], the authors also compared the TVRA from ETSI (described in section 3.4.6.) with TARA. Even recognizing the many similarities, they highlighted that TVRA is mostly designed for data and network communications and it would require a significant level of customization for the automotive sector. The authors of [139] also recommended SAHARA presented in section 3.4.7 of this deliverable to perform the threats identification in the concept design phase of the project (as in the case of DIAS).

Beyond [139], different sources in literature claims that the current description of TARA from SAE J3061 and ISO 21434-2021 is too high level and it should lean on other well defined methodologies. In [137], the authors highlights that existing TARA methods recommended in SAE J3061 just focus on the concept design phase in the lifetime of vehicles and they should be integrated with other methodologies. More specifically for the threat modelling phase, the authors of [137] recommend the integration in TARA of the EVITA methodology described in section 3.4.2 of this deliverable and the HEAVENS security model described in section 3.4.5, which is in turn based on the Microsoft STRIDE model discussed in 3.2.4 and following subsections. For the risk assessment phase, the authors of [137] recommends the integration in TARA of OCTAVE presented in subsection 3.3.3. On the other side, as discussed in section 4.3.3 and in [137], OCTAVE was defined for high level processes in enterprise information security risk assessments but it is not readily applicable for embedded automotive systems.

A similar analysis was conducted in [142] where EVITA, HEAVENS and OCTAVE were recommended for the implementation of TARA even if the limitations or needed customizations of each approach were discussed. SAHARA was also indicated (in combination to STRIDE) for threat identification and analysis.

Then, OCTAVE could complement partially TARA for some steps only. For example, the authors of [140], recommends to use OCTAVE as preliminary risk assessment and use TARA only after. The authors of [140], also highlight that the greatest strength (i.e., its simplicity) of TARA also represents its greatest weakness: it only focuses on the greatest risk, other risks are ignored.

In [141], the authors have discussed how TARA could be enhanced to fulfil its objectives. STRIDE was indicated as a potential and complementary methodology for threat analysis and assessment.

It should also be highlighted that the DIAS project is more focused on the design and development of a prototype (e.g., the ECU demonstrator) but it is not focused on the deployment and operational aspects. Then, the implementation of the TARA in DIAS could be fine for the scope of the project, but it should be extended with some of the methodologies identified in section 3 for future work beyond DIAS in relation to the practical deployment of the DIAS solutions. For example, the authors [137]

recommend for the risk assessment EVITA and HEAVENS in the design phase (e.g., current DIAS project) and the CVSS described in section 3.3.2 of this deliverable for the future operational phase. In fact, in the operational phase, it is common to re-evaluate periodically the risks at the light of new threats and vulnerabilities (e.g., due to the increase capabilities of attacker or obsolescence of security solutions and algorithms)

4 Cybersecurity validation, verification and testing in modern vehicles

4.1 Cybersecurity testing and challenges

As defined in CNSSI-4009 [15], "Security Testing is the process to determine that an information system protects data and maintains functionality as intended". In the testing jargon, the system to be checked is commonly referred as System Under Test (SUT).

Based on the security management and evaluation challenges analysed in D4.1, the implementation of security testing in the automotive context should address at least the following aspects:

- Time, complexity and cost of testing techniques: Common security testing techniques usually require manual interaction from security experts. In IoT, the complexity of the cybersecurity certification could require extensive and expensive efforts, which may delay the product launch and impact negatively the manufacturer from a competitiveness point of view. From the certification point of view, a costly process could imply that the manufacturer cannot afford the costs related to the cybersecurity certification process. On the one hand, the cybersecurity certification could require monetary costs related to the payment to a certification body issuing the certificate, or the testing laboratory being involved in the process. Furthermore, the process can involve qualified personnel to implement the measures required to obtain the certification. This issue is even more relevant in case of update, which may require a re-evaluation and re-certification. Based on this, the cybersecurity testing process should be cost-effective and lightweight to foster its adoption, facing the trade-off between the evaluation assurance level (EAL) and the costs for the companies (time and money), specially for SMEs (Small Medium Entity) and startups. It could also help to deal with security changes, by providing a faster and affordable re-evaluation process.
- Frequent security updates: Cybersecurity is itself a dynamic concept. Security changes can be caused by a new discovered vulnerability, or an update/patch that applies to the certified product. In particular, smart vehicles will be often operating in physical environments with changing conditions that could affect to their security level. Therefore, the security level of the vehicle will change depending on the deployment aspects where it can be exposed to different security attacks. Indeed, at the end of a cybersecurity evaluation process, such system could be evaluated against a set of security requirements, but the initial tested configuration and security level can change [33], [58] due to software updates (which may also be done to address security threats), changes in the system configuration, or the need to be used in different operational contexts. Taking in consideration that software updates and patching may become an important part of the functionality of modern vehicles [123] (this is a function that TESLA already implements[2]), this requires the design of a lightweight and automated cybersecurity evaluation process, in order to reflect the changes introduced in the system and check that no other components' security has been affected. In this sense, the security testing methodology should be able to manage the frequent changes associated to the security level of a vehicle.
- Scalability of testing techniques: The large number of smart vehicles and components to be certified requires the design of cost-effective testing procedures. Ideally, these techniques should be applicable to different types of devices, in such a way that similar procedures could serve to certify the security level of different components. Scalability could also be analysed

[2] https://www.tesla.com/support/software-updates

from a different perspective; indeed, it should be a core aspect of security testing techniques by simulating real-world scenarios in order to detect possible security breaches.

Based on that, the next section 4.2 describes the main security testing approaches that are currently employed on different environments and in which conditions they could be used within the methodology. Before such description, we use different aspects to classify the set of security testing techniques that are considered in this document.

4.2 Taxonomy of cybersecurity testing

Security testing techniques can be classified according to different criteria [73]:

- The knowledge the tester has about the SUT. Whereas in black-box- testing, internal details of the SUT are not used, in white-box-testing, SUT's internal details (e.g., source code) are considered and used to define the tests.
- The execution of the testing procedure, which can be performed manually, so that the human interaction guides the process, or automated, in which the tests are executed or generated through the use of specialised tools
- The procedure to execute the tests can be dynamic, against the running SUT, or static, in which the execution of the SUT is not required.
- The test generation can be performed off-line if all the tests are generated previously to the execution on the SUT, or on-line, in which each test is generated successively after the successful execution of the previous test.
- The focus of the test, which can be the attacker or the SUT.
- The test description, which is related with the test definition language (XML, TTCN, Java, etc.) and linked to the specific testing tool being used [74][75].

The next subsections detail the main testing approaches as well as different techniques or methods associated to each of them. A summarised description of the approaches and their relationship with the criteria can be seen in Table 10.

Table 10 Summary of the testing approaches

Security Testing Technique	Knowledge of the SUT	Execution	Procedure	Generation	Focus
MBT: Behavioral	White-box	Manual/Automatic	Dynamic	Online/Offline	SUT
MBT: Attack patterns	White-box	Manual/Automatic	Dynamic	Online/Offline	Attacker
Regression: Test all	White-box and Black-box	Manual/Automatic	Dynamic	Online/Offline	SUT/Attacker
Regression: Minimization	White-box and Black-box	Manual/Automatic	Dynamic	Online/Offline	SUT/Attacker
Regression: Prioritization	White-box and Black-box	Manual/Automatic	Dynamic	Online/Offline	SUT/Attacker
Regression: Selection	White-box and Black-box	Manual/Automatic	Dynamic	Online/Offline	SUT/Attacker

Manual Code review	White-box	Manual	Static	Offline	SUT
Static application security testing	White-box	Automatic	Static	Offline	SUT
Penetration	White-box and Black-box	Manual/Automatic	Dynamic	Offline	Attacker
Behavioral Fuzzing testing	Black-box	Automatic	Dynamic	Online/Offline	SUT
Data Fuzzing testing	Black-box	Automatic	Dynamic	Offline	Attacker
Vulnerability scanning	Black-box	Automatic	Dynamic	Offline	Attacker
Dynamic taint	Black-box	Manual/Automatic	Dynamic	Offline	Attacker
Mutation-based fuzzing	Black-box	Automatic	Dynamic	Offline	Attacker
MIA evolutionary fuzzing	Black-box	Automatic	Dynamic	Online/Offline	Attacker
Based mutation testing	Black-box	Automatic	Dynamic	Online/Offline	Attacker

4.2.1 Model Based Testing

Model-Based Testing (MBT) [76] is a testing approach based on the usage of models that can represent the test requirements, the SUT itself, and its environment. Based on it, Model-Based Security Testing (MBST) [77] is the MBT variant that validates the security behaviour of a certain software system requirements considering a set of security properties. MBT can be applied to the vehicular domain as discussed in [119],[120].

The model is typically interpreted as a high-level view of the SUT, with high level operations (e.g., sendMsg(valid_id)), and high level types (e.g., ID_TYPE). Therefore, an additional process to turn the abstract model into a specific implementation of the SUT is required. To this end, the technique employs an adapter, which serves as an interface for implementing the real tests that will be run on the SUT. This adapter is mainly composed by two different interfaces: one for the high level operations and one for the high level types. The interfaces, that are usually written in the testing language used (e.g., Java, TTCN3), should be implemented by the tester, linking each high level operation with a specific command/message on the TOE, and each high level type with a specific real value. Therefore, the test suite calls the adapter, which translates the operations and types of each test step, and sends it to the TOE to execute the test step.

The following steps are normally included in the MBT (and MBST) process:

1. Model Design: in this first step, the high level model of the SUT is designed and created from specifications that will drive the specification of testing criteria to evaluate the SUT. Towards this end, we can use formal languages such as UML [78], proprietary languages or Domain Specific Languages (DSL) [79]. The model also specifies the entities and operations that comprise the SUT, as well as their corresponding inputs and outputs.

2. Test Specification: using the previous model, the next step is to define in a high level way the tests we want to implement [121].

4.2.2 Regression testing

The main goal of regression testing [83] is to ensure that any changes to a SUT do not have unintended consequences and that it behaves as expected according to the requirements. When certain specifications change, or the software is updated/patched to meet new requirements, this testing method is required.

This is especially important in the case of security; regression testing is necessary when a new vulnerability is found or when requirements change, which could imply a security patch. The technique can be effectively implemented by choosing the smallest number of suitable tests required to cover a specific change in the security level. This testing mechanism is particularly useful during the operation/update process of the vehicle lifecycle, not only during the manufacturing phase.

In terms of the previously described classification aspects, regression testing may be carried out using white-box or black-box approaches, both in a manual way or with the help of a tool that automates the test selection and/or generation. An example of is the tool proposed by [84], which is an extension of the tool CertifyIt [80],[81]. Depending on the test generation, the method may be on-line or off-line, similarly to MBT [81][82].

Depending on the test coverage, there are many methodologies for regression testing [85]. We can follow a test strategy, in which all the test cases are executed, which may be a costly and time-consuming operation, but other strategies can be used to optimise the testing process to deal with these issues. In this context, Minimisation seeks to reduce the size of a test suite by eliminating redundant test cases from the test suite, for example removing those tests that do not yield coverage of modified or affected portions of the system, or executing only those tests that failed before. Furthermore, Prioritisation is used to order the tests based on a specific criterion, on the basis of history, coverage, or requirements, which is expected to lead to the early detection of faults or the maximisation of some other desirable properties, usually employing mathematical cost functions. Although risk-based approaches are important for testing, there is no approaches using risk for prioritising security regression tests. Finally, Selection is used to pick up a subset of the tests, ensuring that only the parts that have suffered an update are evaluated. Selection techniques are further classified into the subcategories coverage-based, safe and ad-hoc random techniques. Coverage-based techniques select the tests based on identification of the modified parts of the SUT as well as the path and data dependence graph coverage by tests, Safe techniques selects all tests from the previous version which reveal faults in the new system version, and Ad-hoc random techniques consider that all tests of the previous version's test suite may reveal faults and select randomly a certain number of tests [85].

4.2.3 Code-based Testing

Code-based testing [86] is a white-box method for detecting bugs and flaws in the source code in order to catch anomalies early in the development process. Code reviews can either be done manually, in which an expert analyses software code line by line [87], or automatically, which is commonly referred to as Static Code Analysis (SCA) or Static Application Security Testing (SAST) [88]. In the last one, a SAST tool examines a component's code and alerts the user about any possible security issue. It can use syntactic checks, such as calling insecure API functions or using insecure configuration options. It is also possible to use semantic checks that requires knowledge of the program's semantics, such as data and control flows.

SAST software can analyse all control flows of a program in a scalable manner, while manual code reviews are a time-consuming process that needs expertise, experience, and perseverance. These tools will then provide comprehensive guidelines for resolving security problems early in the development process. This technique has a higher coverage of the SUT and a lower false negative rate than dynamic testing approaches (when a test passes but a threat is present). However, SAST tools only disclose vulnerabilities that have been previously defined; therefore, experts must also configure the SAST tool properly in order to detect such vulnerabilities. There are specific tools, which can help in this context: Snik code, LGTM and SonarQube[3].

4.2.4 Penetration Testing

Penetration testing [89] is a testing mechanism to identifying methods for circumventing a SUT's security features by simulating real-world attacks. While this method is usually used to test the SUT's missing functionality or side effects, it may also be used to test the system's environment (e.g., by exploiting an obsolete operating system). Penetration testing may be classified as black- box if the intruder has minimal knowledge about the system, or white-box if internal details are known.

Furthermore, though manual penetration testing is common, there are tools that assist the tester in discovering flaws in a more automated way. This is the case of port or vulnerability scanners [90], which use different techniques to detect security issues in applications. The scanner is used to run a series of predefined attacks against the system's interfaces. The system's responses are then evaluated to see whether the attack was effective, or at least, it was useful to learn more about it and make the attack successful.

There are many penetration testing standards; in this regard, the Open Source Security Testing Methodology Manual (OSSTMM) [91] is the most prominent approach, which offers rules and guidelines for covering the different protocol stack layers. From initial requirements review to the report generation, the OSSTMM approach encompasses the entire risk assessment process involved in a penetration test.

4.2.5 Fuzzing testing and dynamic taint

The concept of fuzzing testing [92] is to test SUT security vulnerabilities by using unintended or incorrect inputs. This method has been shown to be successful in identifying weaknesses that are missed by other testing methods. On the one side, it can be used to test input data (data fuzzing testing [93]) by feeding the SUT with random data to find possible errors or vulnerabilities. On the other hand, it can be used as behavioural fuzzing testing [94] [95], in which valid/invalid message sequences are used with the same purpose.

Since fuzzing testing does not need knowledge of the SUT implementation details, it is referred to as a black-box testing process. Behavioural fuzzing testing, like in MBT and penetration testing, can be on-line or off-line, depending on the test generation. Regarding data fuzzing testing, the followed approach is typically off-line, since tests are generated before they are executed.

A fuzzing technique is based on a fuzz generator, also known as fuzzer, which is the algorithm in charge of randomising using different strategies. The most basic form of fuzzing testing is the random fuzzing [93], in which data for testing is randomly generated. Other, more sophisticated methods are the mutation- based fuzzing testing [96], the model-based mutation testing [97], the model inference supported (MIA) evolutionary fuzzing testing [98], or the method suggested by [99], which combines MBT and fuzzing techniques.

[3] SAST tools. https://snyk.io/blog/sast-tools-speed-comparison-snyk-code-sonarqube-lgtm/#:~:text=Static%20Application%20Security%20Testing%20(SAST,14x%20times%20faster%20than%20LGTM

In mutation-based fuzzing, the fuzzer has some knowledge of the SUT's input format, so the mutation-based fuzzing tool will create new variants based on existing data samples (mutants). In the model-based mutation testing approach, the attack model is mutated by choosing different attack paths. Finally, MIA evolutionary fuzzing incorporates model inference and a genetic algorithm to exploit a SUT's possible cross-site scripting vulnerability. Since the tests are generated based on the attacker model (on-line or off-line), this technique is similar to MBT. Despite the high efficacy of this technique and its ability to detect zero-day vulnerabilities, each SUT needs its own fuzzer. Furthermore, the number of generated tests must be limited by a specific parameter or property. These factors may pose significant scalability challenges, especially in complex systems with a wide number of components, as in the vehicular one.

As a way to enhance fuzzing testing, the aim of dynamic taint is to label specific data (e.g., coming from an untrusted source) as tainted. Then, the way the SUT employs the tainted data can be used to recognise insecure data flow. Unlike static analysis, which focuses on identifying problematic data flows, dynamic taint analysis is carried out in the background while the SUT is running. This technique can be combined with fuzzing testing to gain information about path execution by choosing the most promising test sequences to detect possible vulnerabilities [100].

4.3 Projects, standards and initiatives within the vehicular context addressing testing

4.3.1 ISO 26262

The functional safety standard ISO 26262 [68] "Road vehicles – Functional safety" covers electrical and electronic automotive systems and the development process, including requirements specification, design, implementation, integration, verification, validation, and configuration, as shown in Figure 6. Regarding testing, chapter 6 of the ISO 26262 specification contains guidelines for unit and integration testing, and recommendations in Table 10 contains all software unit testing methods (for a more detailed analysis on software testing methods, look also to [69]). In this regard, the standard considers different testing methods: requirement-based testing (mandatory for all ASIL ratings A to D) in which in- put values are derived from the behavioural requirements; interface testing, to detect early failures that could go unnoticed till the integration testing process (mandatory from ASIL A to D); fault injection testing (mandatory for ASIL D), which entails the insertion of certain arbitrary faults (e.g., code mutation, corrupting the variable value, etc.) either at compile or run time; resource usage testing to ensure that the unit being tested does not consume excessive amounts of resources (mandatory for ASIL D); and back-to-back comparison testing, aiming to validate that the SUT behaves as the model based on which it was designed (mandatory for ASIL C and D). Despite the recommendations, the standard does not give indications on how to automate the testing process and moreover, the tests are focused only on safety aspects.

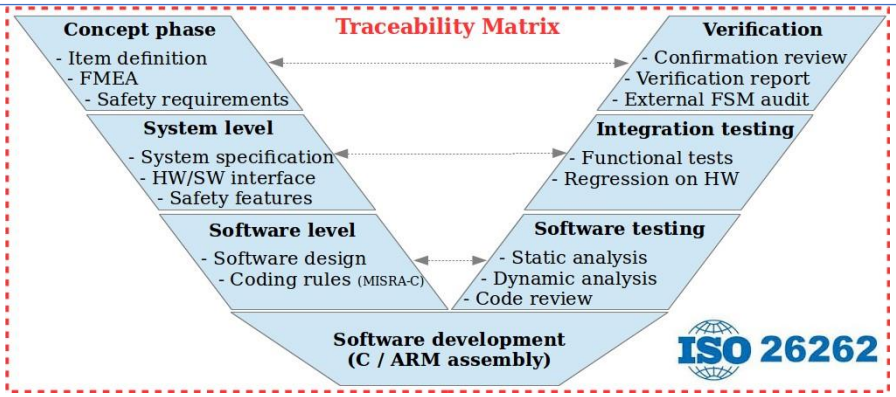


Figure 6 ISO 26262 traceability matrix

4.3.2 SAE ISO 21434

The ISO/SAE 21434 [124] safeguards the entire development process and lifecycle of a road vehicle and promotes "security by design". Following the V-model in a similar way to ISO 26262, it includes requirements engineering, design, specification, implementation, test and operation. The ISO/SAE 21434 is therefore a process-oriented standard and helps define a structured process to ensure cybersecurity along the lifecycle, so it does not prescribe specific cybersecurity technologies, solutions or remediation methods.

ISO 21434 applies to the cybersecurity relevant items and components of a series production road vehicle including aftermarket and service parts. Systems external to the vehicle can be considered for cybersecurity purposes but they are not in the scope of the standard.

ISO/SAE 21434 is based on an overall risk management methodology, which is applied to all the supply chain phases in the vehicle production including the manufacturer organization itself.

As with other standards described in this section, the ISO standard is mostly focused on cybersecurity rather than tampering.

While ISO/SAE 21434 follows a general risk management approach, it is linked to the specific production activities in an organization including the V-model for production development as described in section 10 of ISO/SAE 21434.

4.3.3 J3061

SAE J3061 [101] officially published in January 2016, is considered as the first standard addressing automotive cybersecurity. It provides a set of high-level cybersecurity principles and guidance for cyber-physical vehicle systems. The guiding principles of J3061 are similar to other cybersecurity standards with an emphasis on understanding the security risks through threat analysis during the earliest stages of development. The main objectives of J3061 and the related TARA methodology is to manage these risks and implementing best practices throughout concept, design, and implementation and testing.

In all cases (hardware and software level), the standard considers a specific process of testing. In particular vulnerability and penetration testing, as well as unit and integration testing are considered to guarantee that all the cybersecurity requirements established during the design phase have been met by the system. Although SAE J3061 does not explicitly require software tools and automation, the reality is that it would be difficult for developers to comply with the guidelines without tool support. Indeed, the standard lists in the annex some testing tools that can be considered, such as static Code Analyser, Encryption Cracker, Dynamic Code Analyser, hardware Debugger, Network Traffic Analyser,

Known Answer Tester, Vulnerability Scanner, Application Tester, Fuzz Tester, Interface Scanner, Exploit Tester and Network Stress Tester.

4.3.4 Best Practice Guide for Cybersecurity and Intelligent Transportation Systems provided by USDOT

Even if it is not a regulation, the US transportation department (USDOT) provided a report about best practices for automated vehicles cybersecurity focused on penetration testing [102]. The primary focus of the report is on safety and the integration of vehicles with the existing road infrastructure and other modes of transportation.

The report manifests the importance of penetration testing to provide indications about the feasibility of attacks exploiting the target vulnerabilities and the effectiveness of current security protections. Furthermore, it can be also used to estimate the likelihood or the potential impact. The scope of the penetration tests can include security policies, devices, applications, networks, access controls, communications and configurations that comprise the locality ITS infrastructure as well as interfaces to external systems or Regional ITS managed by other entities. As not always it is possible to test everything, the report advises to focus the penetration test on the most critical functions and the implementation in the ITS, taking into account the NISTIR 8179 Criticality Analysis Process Model: Prioritising Systems and Components.

4.3.5 ETSI

Based on IEEE Std 1609.2-2016 "IEEE Standard for Wireless Access in Vehicular Environments Security Services for Applications and Management Messages", ETSI TS 103 097 specifies a security header and the certificate formats for Intelligent Transport Systems. These formats are defined specifically for securing G5 communication (V2V). Based on that, ETSI TS 103 096 defines a test suite to conform to the formats specified in the standard. Although the tests are focused specifically on verifying that functionality and conformance, ETSI has made available the test suite [103] implemented in Testing and Test Control Notation Version 3 (TTCN-3) format, with the objective of facilitating the verification and testing of the standard.

4.3.6 PEGASUS

The research project PEGASUS (Project for the Establishment of Generally Accepted quality criteria, tools and methods as well as Scenarios and Situations on the release of highly-automated driving functions) was promoted by German Federal Ministry for Economic Affairs and Energy (BMWi) for the establishment of generally accepted quality criteria, tools and methods as well as scenarios and situations for the release of highly-automated driving functions. In particular, the PEGASUS project has defined a comprehensive framework for the verification and validation of automated vehicles.

PEGASUS considers a specific phase of testing that comprises test generation though logical test cases and test execution, enabling the possibility to execute the single test cases on different test tools, such as simulation or real world tests. The testing is handled as black box, without knowing the internal details of the system. After testing, PEGASUS considers an evaluation phase, whose purpose is to confirm the compliance of the test object with pre-defined behavioural criteria (e.g., keeping appropriate safety distances, not causing collisions, etc.). However, the focus is on functional and safety tests.

4.3.7 Other approaches

At research levels, we can also find some interesting approaches that address testing in the automotive context. Although this review will not be exhaustive, the main objective is to present a

brief overview of the testing research lines and preferences (in terms of testing techniques) of the community. A more detailed analysis of the current state of the art in research can be found in [108].

In [109], the authors review the main phases of the SAE J3061 standard, suggesting approaches to instantiate them. In particular, they propose attack trees to deal with the vulnerability analysis phase, fuzzing testing for discovering functional failures in the inputs and grey-box penetration testing for security testing. However, the paper is mainly theoretical, so it lacks of a real application of the suggested techniques. In [110], the authors develop a security-by-design framework for AV, in which testing and risk assessment is considered as part of the cybersecurity management. Although they give some options regarding testing techniques (E.g., ethical hacking, penetration, fuzzing and network testing), the approach is at a high level, and they just describe these mechanisms. Authors in [111] outline a structured testing process based on ISO/SAE 21434 for verifying and validating automotive cybersecurity, for which there is no standardised method so far. The process is flexible in order to allow implementers to utilise their own, accustomed tool-sets. The framework combines testing and risk assessment to prioritise the tests and to decide if the test result is acceptable given a threshold. The testing methods considered are functional, interface, penetration, and fuzzing testing, as well as static code analysis and vulnerability scanning. Authors in [112] propose an evaluation cybersecurity system for vehicles. the paper mainly focuses on the risk assessment, combining CVSS with EVITA, HEAVENS and SAE J3061. Although testing is part of the evaluation process, the details about how it is performed, the testing technique followed and which tests have been taken into account are not present.

In [113], the authors propose the framework of a testbed that can be used to simulate cyber attacks on virtual vehicle running in a connected environment.

The tool is intended to evaluate the effectiveness of designs against specific cyber attacks. In [114] authors perform security testing using a test-bed that simulates the behaviour of the real vehicle. In particular, they validate its approach through an On-Board Diagnostics (OBD) based attack, acting as the attacker against the test-bed. Regarding fuzzing testing, [115] analyses the current state of the art in the automotive context. In particular, authors claim that current publications focus more on the configuration of fuzzing tools than on the usage of fuzzing in the automotive context. Therefore, they provide an analysis of the main characteristics and limitations of fuzzing testing, and design a CAN fuzzer able to randomise the format of the CAN packets while monitoring the results of the inputs.

Using a model based testing approach, authors in [116] test Over-The-Air (OTA) updates automating both the testing generation and execution. The attacker objectives and steps are modelled in attack trees and the sequence of the test steps are automatically derived from it. After linking each step with a script, the tool is able to execute the different steps over the testbed, which is based on the UPTANE reference framework for software updates.

4.3.8 Discussion and analysis of the testing approaches.

Coinciding with the analysis performed in [118], we can see that currently, penetration testing leads the testing strategies to evaluate the security of a vehicle system. However, although this technique is important for vulnerability analysis and successful to a certain degree, it is time consuming and costly, as it is mainly manual. Fuzzing testing provides a more automated way of testing through fuzzers. It is capable of generate thousands of tests with hardly any effort. However, its usage requires knowing the format of what we want to randomise and it is mainly limited to the analysis of data inputs. In the same line, Code Based testing also provides an automated testing techniques if used with adequate tools. Its usage requires access to the source code, as it is intended to be performed when the SUT is being developed, but it provides a high coverage with a very low false negative rate.

Its main limitation is that is only able to discover vulnerabilities previously defined by the tester, whereas fuzzing and penetration testing can discover new vulnerabilities. Regression testing appears as a complementary testing technique dealing with the changes on the SUT, and ensuring that it is still behaving as expected. However, this technique has to be complemented with other testing techniques, as it only addresses the way the set of test is selected to verify the TOE functionality after a change. Finally, MBT represents a promising approach for the security testing in the automotive domain, due to the possibility of automating the generation of the tests, and therefore, dealing with security changes and scalability aspects [117]. The automation of the test generation and execution, and the simplicity of modelling the TOE at a high level, are the main reasons to consider MBT for the instantiation of the security testing process. Although the implementation of an adapter is required, further modifications and repetitions of the tests do not require significant changes of the adapter. However, there is no a silver bullet and a vehicle security testing approach could embrace different techniques to deal with the security evaluation challenges during the whole life cycle of the system.

4.3.9 Relationship between the landscape analysis of the testing methodologies and the actual methodology used in the DIAS project

Testing of automated vehicles for cybersecurity and more specifically for tampering is still in an early phase, but it will probably grow in importance in future years as vehicle will become increasingly sophisticated and with increasing levels of connectivity and automation. Because the vehicles are becoming more and more dependent on their internal computing and connectivity capabilities, techniques and methods developed for cybersecurity testing [70] can be applied and customized for tampering attacks and methods, at least when computing platforms (e.g., ECU) and connectivity systems (e.g., CAN-bus in-vehicle network) are investigated and under test. Most of the techniques described in the previous subsections of this section 4 have been applied in DIAS for different types of tests.

5 Report on DIAS demonstrator security testing

5.1 Introduction

Based on the security analysis and requirements identification performed in Task 4.1 (Ref. D4.1) [125] and the market assessment of cheating devices performed in Task 3.1 (Ref. D3.1) [126], three categories of the security are defined as critical, as shown in Figure 7:

- Category 1 is the Engine Control Unit (ECU) reprogramming.
- Category 2 is the CAN-bus communication between the ECU and the sensors.
- Category 3 is the data sending to the cloud.

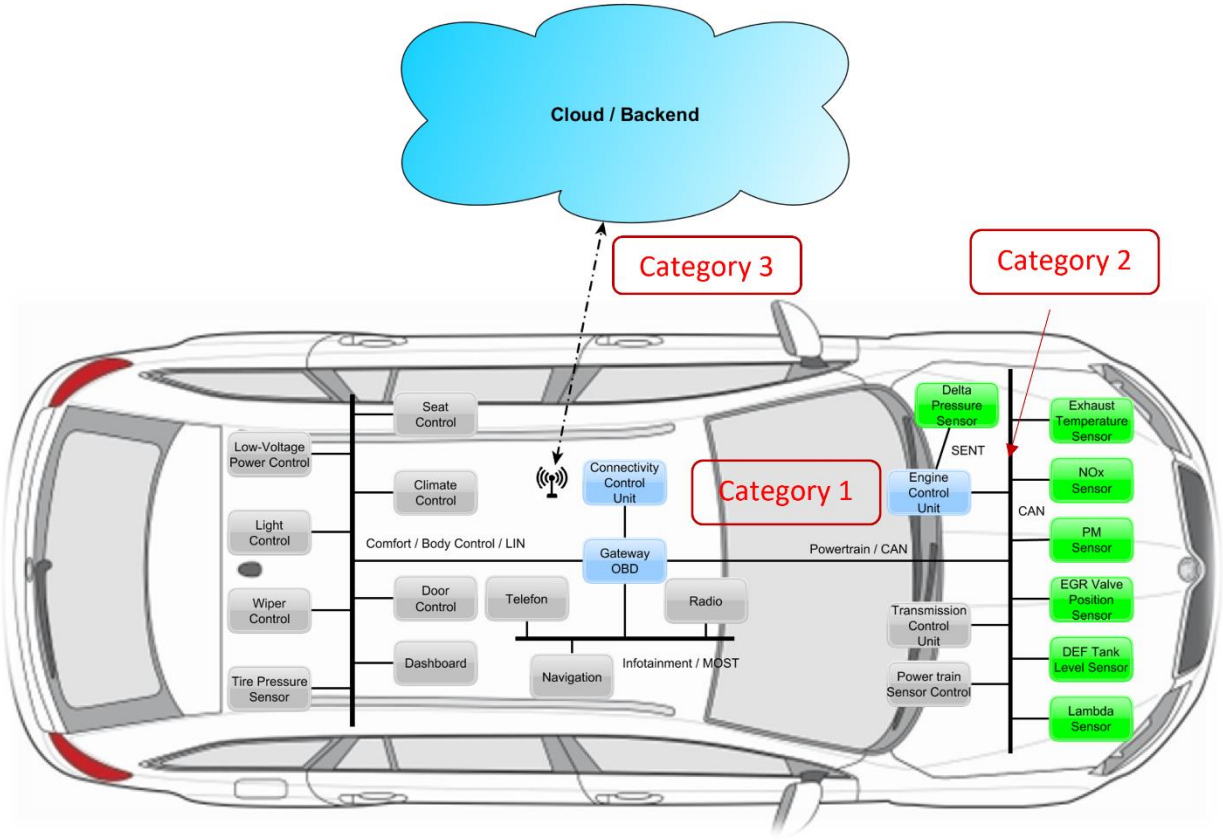


Figure 7 Simplified generic vehicular architecture and three categories of security

In Tasks 4.2 and 4.3, several security solutions are developed covering all the three categories to ensure the security of the whole system. In the 1st category, secure boot, secure software update, code signing solutions are applied on the ECU software to protect the authenticity and integrity during the software flashing operation. For the 2nd category, secure CAN communication between the engine control units and the sensors is ensured using lightweight Secure Onboard Communication (SecOC Light) and several prototype secure communication solutions are also developed and demonstrated, such as key distribution and management schemes, data authentication approach, secure certificate transfer, firewall and intrusion detection systems (Ref. D4.2) [127]. For the 3rd category, the blockchain technology is employed to provide tamper-proof cloud connection (Ref. D4.3) [128].

Within Task 4.4 these implemented countermeasures and prototype security concepts are verified and validated using penetration testing techniques and concept review. In this chapter the results of penetration testing will be presented, in Chapter 6 the concept review of a key exchange scheme for

sensor control units (SCU) will be reported. Further, another design review and code review of the 3rd category security solutions were conducted in T4.4. But due to the contents of sensitive vulnerability findings the results will be documented in a separate report.

For the penetration testing two sets of test environments are used. The desktop test setup is used as much as possible for testing the authenticity and integrity of ECU software reprograming and of the communication between ECU and SCU. The desktop setup is also used at the early phase as tool to understand the system and to test scripts. However, the ECU and software, which are used on the desktop setup does not support reading out the error flags via INCA (Integrated Calibration and Application Tool) [130] and all Diagnostic Trouble Codes (DTCs) are disabled as the rest bus is not available. If they are not disabled, there will be too many errors and the related errors for the targeted attack cannot be registered due to the limitation of error memory. It was attempted to modify the software to support partial DTC registration, but it was not successful. Due to this reason the tests for the secure communication between ECU and SCU cannot be verified using the desktop setup due to lack of error information. For this reason, on-vehicle tests are executed with a development ECU which has the interface to access the error flags via INCA.

An overview of the executed tests in scope of Task 4.4 is shown in the following table.

Table 11 Summary of penetration test activities in Task 4.4

Type of the attack	Attack target	Description	Platform
Brute force attack (Subsection 5.2.1)	ECU reprogramming	The test is to brute force the secure access service (0x27) of Unified Diagnostic Services (UDS) to verify the countermeasure of unauthorized ECU reprogramming	Desktop test setup
Tampering attack (Subsection 5.2.2)	ECU reprogramming	The test is to tamper the flash image and test the integrity check of ECU software during reprogramming	Desktop test setup
Man-in-the-middle attack (MITM) (Subsection 5.3.1 and 5.3.2)	ECU+SCU (SecOC light)	This group of tests is to test the authentication and integrity of the communication between ECU and SCU.	Desktop/On-vehicle test setup
Fault frame injection attack (Subsection 5.3.3)	ECU+SCU (SecOC light)	This group of tests is to test the authentication and the integrity of the communication between ECU and SCU.	Desktop/On-vehicle test setup
Replay attack (Subsection 5.3.4)	ECU+SCU (SecOC light)	The test is to test the authentication of the communication between ECU and SCU.	Desktop/On-vehicle test setup
Circuit modification (Section 5.4)	Analogue signal of NOx Sensor	This test is to check the risk of analogue signal tampering, which causes integrity issue.	Desktop test setup
Flash dump (Section 5.5)	Memory of NOx SCU	This test is to check the possibility of the memory dump through hardware pins, which can be an attack path to breach the integrity of the system.	Desktop test setup

In the following sections, section 5.2 describes the penetration test results of ECU reprogramming, section 5.3 evaluates the penetration test results of communication between ECU and SCU, section 5.4 shows the analogue signal tampering of NOx Sensor and section 5.5 is for unauthorized memory access of SCU, finally in section 5.6 the results of all tests are summarized.

5.2 ECU reprogramming penetration test

In the DIAS Task 4.1 the threat analysis and risk assessment (TARA) was performed, and the threats of the unauthorized reprogramming of ECU software and data were identified. By mounting this attack, the intruder will be able to make the ECU run unauthorized software or flash falsified/malicious data and may lead to reporting of false emission values by the ECU which will go undetected. The security level of these two threats were rated as critical, and therefore should have high priority to be protected. Accordingly, the generic high-level security requirements of general control unit and ECU were derived (see D4.1 Section 5.2). It requires that secure software update with the help of integrated Hardware Security Module (HSM) must be supported in ECU. On the other hand, from the market assessment of the tampering devices in Task 3.1 (Ref. D3.1, D3.2) [126] [131], the unauthorized ECU reprogramming is the tampering method which is most likely to be used for altering or deactivating the environmental protection systems.

Within DIAS, the secure software update and code signing have been implemented in the MD1 ECU using the Code Verification Certificate (CVC) with Elliptic Curve Cryptography (ECC) P256 – NIST curves [132]. The ECC P256 is an approach of public-key cryptography where the Elliptic Curve Digital Signature Algorithm (ECDSA) is employed for digital signature. The advantage of the ECC P256 is that it delivers full 128-bits security level with a relatively small key size, reducing storage and transmission requirements. In the Figure 8, the signature generation and the verification between the ECU and a tester which is used for ECU reprogramming are illustrated. A tester wants to gain access to the ECU for software reprogramming and it sends requests for a challenge to unlock the ECU, the ECU then send a security challenge to the tester, the tester will use its own private key to generate a signature. The signature is sent back to the ECU and the ECU will use the public key to verify the signature. Once the signature is successfully verified, the ECU sends a positive response, and the ECU is unlocked. The tester can run the next steps for the software and dataset reprogramming.

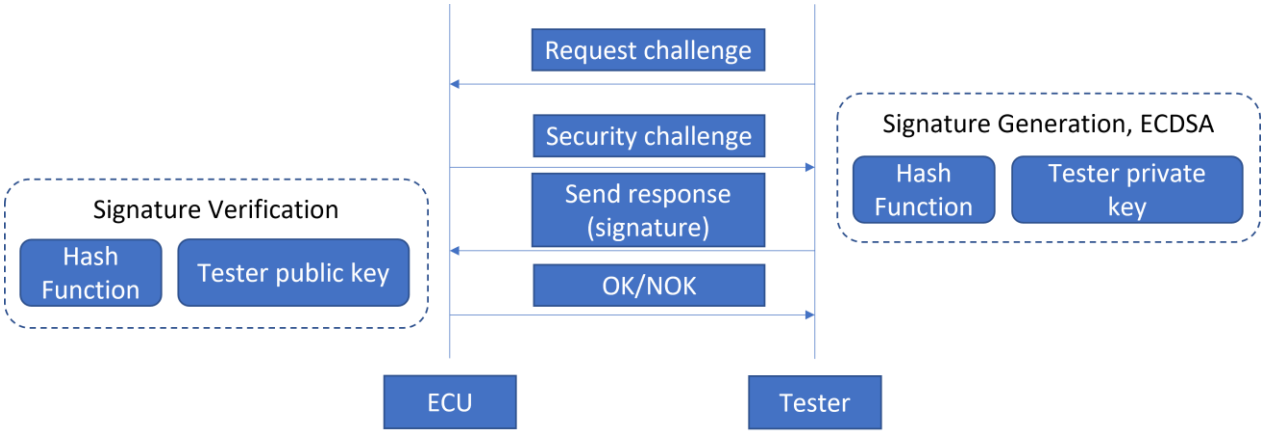


Figure 8 Security Access of ECU reprogramming

The test cases in this group have been designed to validate this secure software update mechanism. There are two categories of the test cases:

- Unauthorized software and dataset reprogramming: brute force attack on the security access

- Programme and dataset modification: tampering attack targeting/affecting the software and dataset to be flashed

5.2.1 Unauthorized software and dataset reprogramming

5.2.1.1 Information gathering

In the scope of this test, an ECU with activated SecurityAccess service (UDS service 0x27) is given as target. As shown in the Figure 9, a development ECU is connected to a test computer where a Raspberry Pi 4 equipped with CAN adapter is used. The test computer works as a tester for flashing the software and dataset on the ECU.

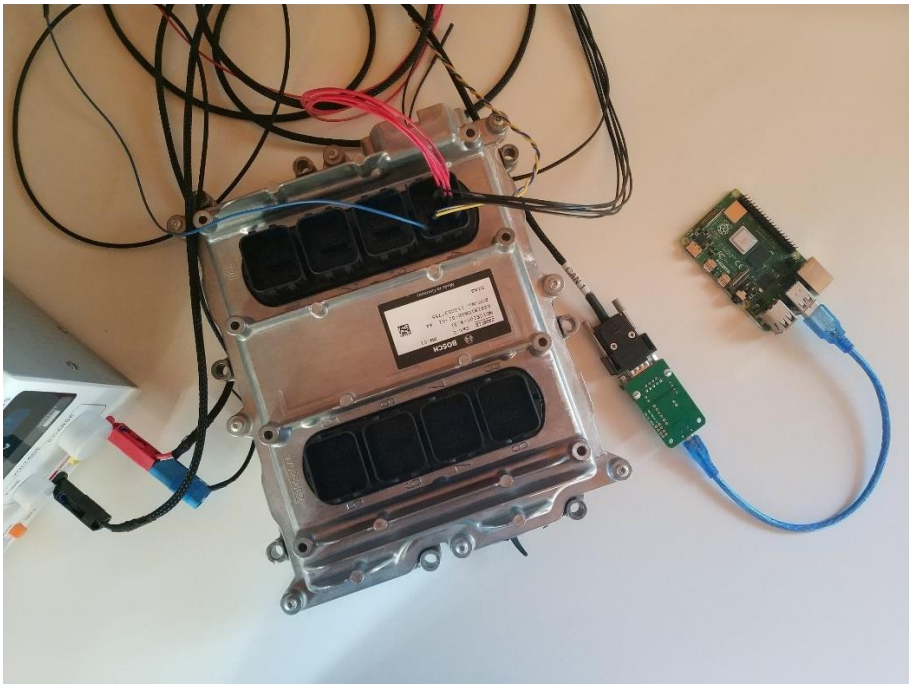


Figure 9 ECU reprogramming penetration test set up

With the open-source tool Caring Caribou [129] the relevant CAN ID of the ECU is found. The communication between ECU and the raspberry Pi can be verified by sending UDS service readDataByIdentifier (0x22). The Raspberry Pi 4 can receive the positive response from the ECU.

In this test group, brute force attacks are performed to validate the authorized access of ECU for secure software update.

5.2.1.2 Test execution

Test case 1*:* as shown in Figure 8, first the test computer sends data to the ECU to request the challenge, which is a data trunk as seed for calculating the signature. A challenge is received from the ECU. According to the provided information, the signature shall be calculated based on the received challenge. The first brute force attack is to send a random signature guess without requesting a new challenge. If the attacker can repeatedly try this attack until the correct signature is found, the secure access is broken. The test result is shown in Figure 10, after the first try a negative response 0x35 is received. 0x35 means *invalidKey*, which shall be sent if the value of the key (in this application it is the signature) does not match the server's internally stored/calculated key. At the second try a negative response 0x24 of service 0x27 is received. The negative response 0x24 is a *requestSequenceError*, which shall be returned if the 'sendKey' sub-function is received without first receiving a request message. So, a 'sendKey' sub-function is not allowed to call without first to invoke a challenge-request sub-function.

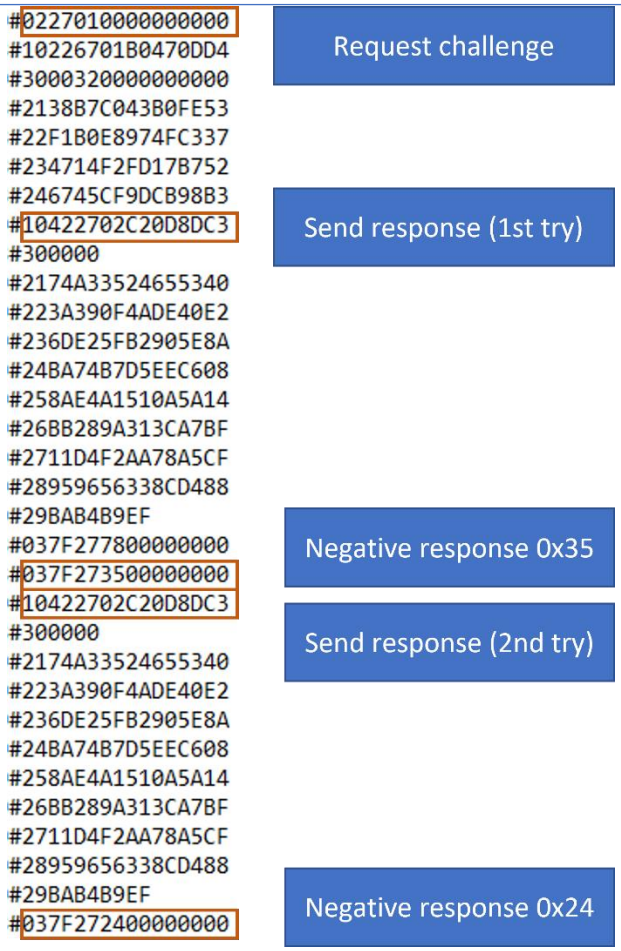


Figure 10 CAN log for sending key without requesting new challenge

Test case 2: in the second test the challenge is requested every time before a 'sendKey' sub-function is invoked. Then a random signature guess is sent back to the ECU. This trial will be repeated as many times as required until the ECU is unlocked or the limitation of repeating is reached. However, at the third failed try, a negative response 0x36 of service 0x27 is received. The negative response 0x36 is an error of *exceededNumberOfAttempts*. It shall be sent when the delay timer is active due to exceeding the maximum number of allowed false access attempts. In this test, the maximum number of attempts is two. When the attempt exceeds this number, the negative response will be received. Due to the length of the signature (256 bits), the chance to guess the signature correctly just with one attempt (as the challenge is newly requested every time) is extremely low.

5.2.1.3 Conclusion

As evidenced by the test result above, the target ECU is secured by setting request sequence error and allowed maximum attempts against brute force attack of UDS service 0x27.

5.2.2 Program and dataset tampering

5.2.2.1 Information gathering

The target ECU and the test set up is the same as in the last test group, which is shown in Figure 9. An integrity check after the software and dataset flashing is implemented in the ECU to protect against unauthorized software and dataset content change.

This tampering test aims at validating the protection mechanism for ensuring the integrity of software and dataset.

5.2.2.2 Test execution

Test case 3: A script is written to flash the software and data onto the ECU. In the script additional codes are added to modify a tiny set of the original software and dataset randomly. The result is shown in Figure 11, after flashing the software a negative response 0x72 of service 0x37 (RequestTransferExit) is received. The negative response 0x72 is a *generalProgrammingFailure*, which indicates that the server detected an error when erasing or programming a memory location in the permanent memory device. This test cannot pass the memory check of ECU.

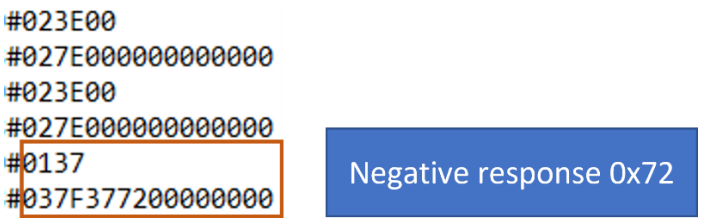


Figure 11 CAN log for failed memory check

5.2.2.3 Conclusion

Without the knowledge of checksum calculation method, the software and dataset cannot be tampered. The software and dataset integrity of the target ECU is protected by the memory check. It should be mentioned that the integrity attack can be implemented only following a successful security access of service 0x27, which has been validated in the Section 5.2.1.

5.3 Secure CAN communication penetration test

As introduced at beginning of this chapter, the 2nd category of security is the secure CAN communication between the ECU and the SCU. This threat has been identified in the TARA conducted in Task 4.1 where the attacker can replace the SCU with a malicious device and send falsified CAN messages which cannot be differentiated by the ECU from normal messages that the original sensor sends. In the DIAS D4.1 [125] Section 5.4, the requirements of secure communication over CAN are defined. The requirements “Communication over the CAN bus must be protected through authentication” and “Communication over the CAN bus must be protected for integrity” are tested in this group of test cases.

In the scope of this test, an ECU and a NOx Sensor (i.e., the SCU) are given as targets. The communication between them is protected by SecOC Light (Secure Onboard Communication Light) (Ref. D4.2) [127]. The main concept of SecOC Light is illustrated in Figure 12. The ECU generates a random number and sends it to the SCU. From that time on SCU and ECU record only the significant data for a defined time. Then, the SCU is assumed to send data to the ECU cyclically. Both sender and receiver calculate afterwards a Message Authentication Code (MAC), using the recorded data and the current random number, by using the same private key. The SCU will then send the MAC frame to the ECU, and the ECU will compare the calculated MAC to the received MAC.

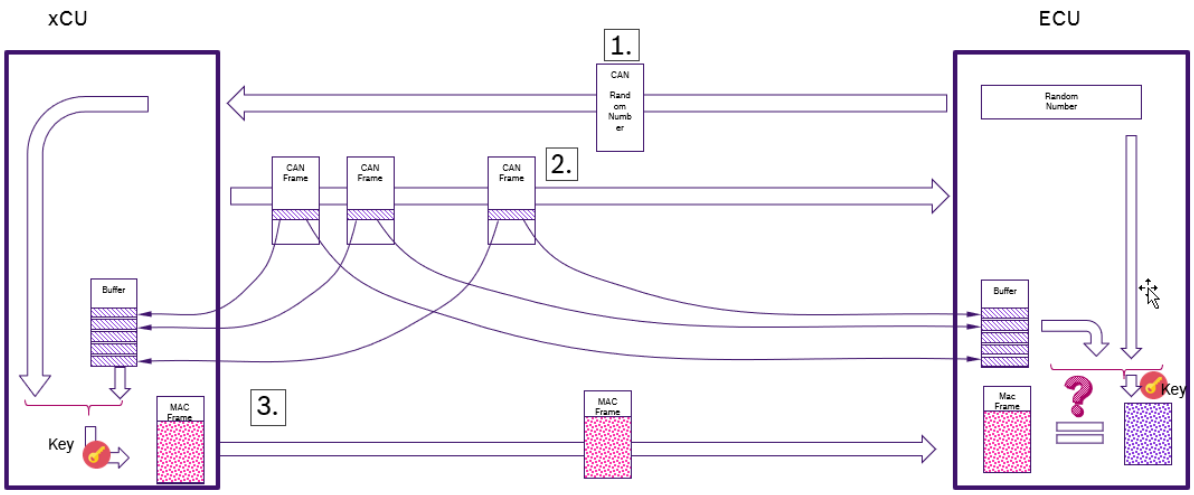


Figure 12 SecOC Light authentication concept (source: DIAS D4.2)

Within this scheme the random number is introduced to protect communications against replay attacks. In this concept, it is foreseen that a fresh random number is generated after each MAC frame. Sensor data tampering is marked as number 2 in the Figure 12, the results are shown in Section 5.3.1. The MAC tampering is marked as number 3 in Figure 12, the result is shown in Section 5.3.2.

The desktop hardware setup of this test group is shown in Figure 13, where the number 1 marks the Raspberry Pi 4, the number 2 is the MD1 ECU, the number 3 points out the NOx SCU, the number 4 is the CAN adapter, and the number 5 is the NOx sensor.

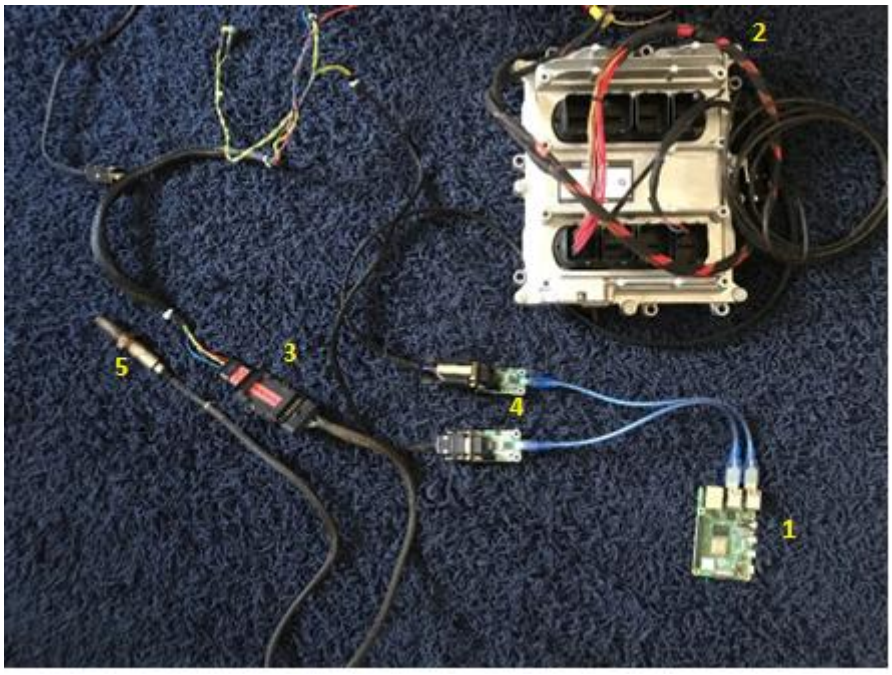


Figure 13 Secure CAN communication desktop test setup

The tests of this group are first executed on desktop test setup. Due to missing of other components of the vehicle on the desktop setup, the software is calibrated to disable the logging of DTCs. So, no DTC can be read out in the error memory. The same tests are executed again on the vehicle. Figure 14 illustrates the test setup on vehicle. On the left side is the laptop with INCA installation and on right

side is the Raspberry Pi which is connected to the vehicle CAN. With the help of the development ECU on vehicle, INCA can be used to check the error flag values of the software.

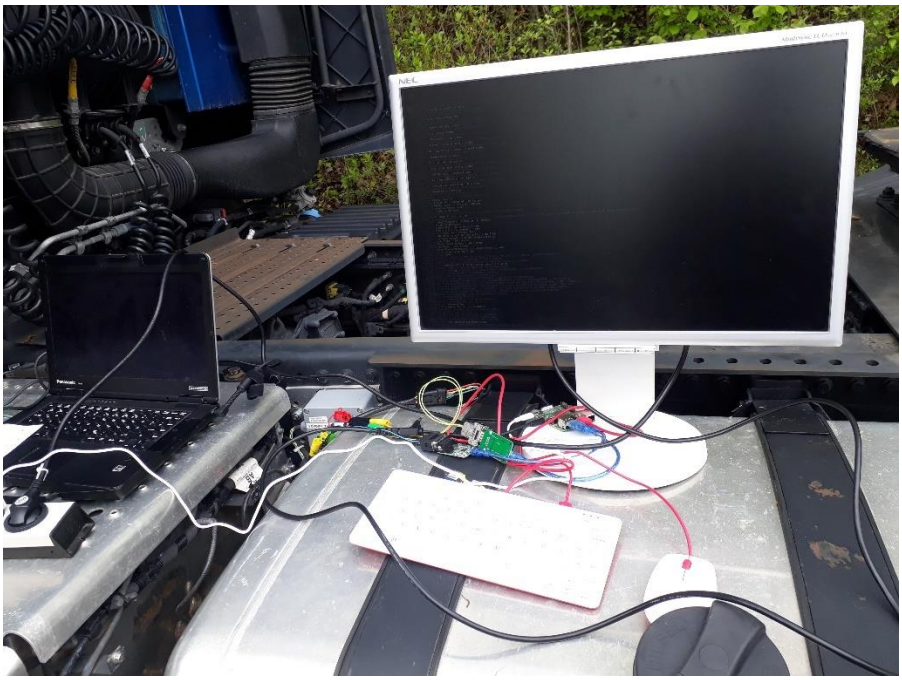


Figure 14 Secure CAN communication on-vehicle test setup

5.3.1 Tampering of sensor data

5.3.1.1 Information gathering

The sensor data is sent with message which has fixed CAN-ID to ECU. The random number and MAC are also sent with messages which have fixed CAN-ID.

This test is to test the data integrity of transmitted data on CAN bus. The MITM attack is used in this group of tests. MITM attack is a cyberattack where the attacker secretly relays and possibly alters the communications between two parties who believe that they are directly communicating with each other, as the attacker has inserted themselves between the two parties.

5.3.1.2 Test execution

Test case 4: First the setup, which is illustrated in Figure 15, is configured to allow a MITM attack. In this setup the communication between ECU and SCU is disconnected. The CAN messages from SCU are redirected to CAN0 interface of the Raspberry Pi and the messages from ECU are redirected to the CAN1 interface of the Raspberry Pi. A script is written to tamper and bypass the corresponding messages. In this test only one byte of the sensor data is tampered, and the other messages are bypassed. With the help of INCA, the error flags in ECU software are checked. From our test result, it can be proved that the ECU can detect the tampering of sensor data by MITM attack as shown in the details presented in Figure 16.

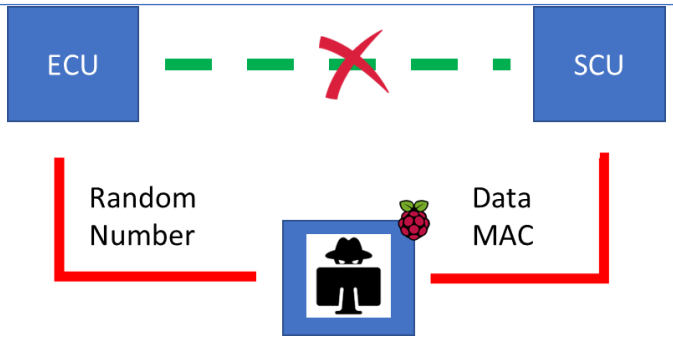


Figure 15 Illustration of MITM attack

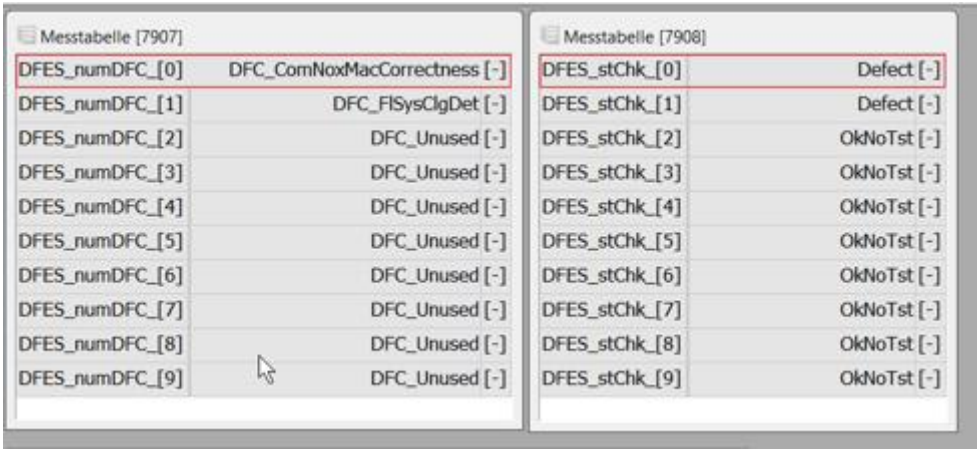


Figure 16 Error flags when data tampering is detected (red marked labels)

5.3.1.3 Conclusion

The data tampering with the man-in-the-middle attack was detected by the ECU. The secure mechanism against sensor data tampering without knowledge of the MAC calculation method and the key is sufficient.

5.3.2 Tampering of MAC data

5.3.2.1 Information gathering

The MAC on the sensor side is calculated with the received random number from ECU and the measured sensor data. It is sent with fixed CAN-ID and fixed interval. The MAC is used to authenticate the communication.

5.3.2.2 Test execution

Test case 5: The same hardware setup as the test of the tampering sensor data is made. The Raspberry Pi acts as “man-in-the-middle” device. The sensor data is not modified and forwarded to the ECU, but the first byte of the MAC is tampered and send to the ECU.

If the tampering is detected by ECU, the corresponding flag is set. With INCA the setting of the flag is checked. As it is shown in Figure 17, the tampering is detected by ECU.

Messtabelle [7907]		Messtabelle [7908]	
DFES_numDFC_[0]	DFC_ComNoxMacCorrectness [-]	DFES_stChk_[0]	Defect [-]
DFES_numDFC_[1]	DFC_FlSysClgDet [-]	DFES_stChk_[1]	Defect [-]
DFES_numDFC_[2]	DFC_Unused [-]	DFES_stChk_[2]	OkNoTst [-]
DFES_numDFC_[3]	DFC_Unused [-]	DFES_stChk_[3]	OkNoTst [-]
DFES_numDFC_[4]	DFC_Unused [-]	DFES_stChk_[4]	OkNoTst [-]
DFES_numDFC_[5]	DFC_Unused [-]	DFES_stChk_[5]	OkNoTst [-]
DFES_numDFC_[6]	DFC_Unused [-]	DFES_stChk_[6]	OkNoTst [-]
DFES_numDFC_[7]	DFC_Unused [-]	DFES_stChk_[7]	OkNoTst [-]
DFES_numDFC_[8]	DFC_Unused [-]	DFES_stChk_[8]	OkNoTst [-]
DFES_numDFC_[9]	DFC_Unused [-]	DFES_stChk_[9]	OkNoTst [-]

Figure 17 Error flags when the tampering of MAC is detected (orange marked labels)

5.3.2.3 Conclusion

The MAC tampering with the man-in-the-middle attack was detected by the ECU. The secure mechanism against unauthenticated data transmission and data integrity attack is sufficient.

5.3.3 Fault frame injection attack

Fault injection is a testing technique for understanding how computing systems behave when stressed in unusual ways. In this group of tests, the tampered frames are injected to test the authenticated communication and integrity of the data frame.

5.3.3.1 Information gathering

In this group of tests, different frames with or without tampering are injected into the CAN bus communication between ECU and sensor. The data integrity of transmission and authentication of the communication are tested. The Raspberry Pi acts in these tests as a node of CAN bus. The hardware setup is illustrated in Figure 18.

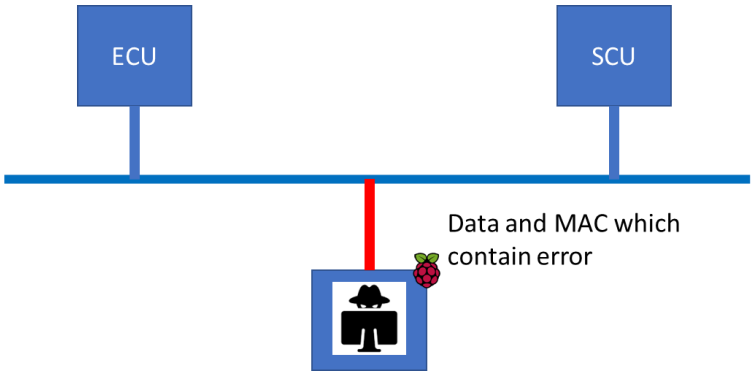


Figure 18 Illustration of frame injection attack

5.3.3.2 Test execution

Test case 6: the Raspberry Pi is used to sniff the CAN bus communication. If the random number frame or the MAC frame is received, they are to be injected back again to the CAN bus without modification. The goal of the test is to check if double sent entries cause an error. The test result in Figure 19 shows that the software tolerates the double sent entries and report no error.

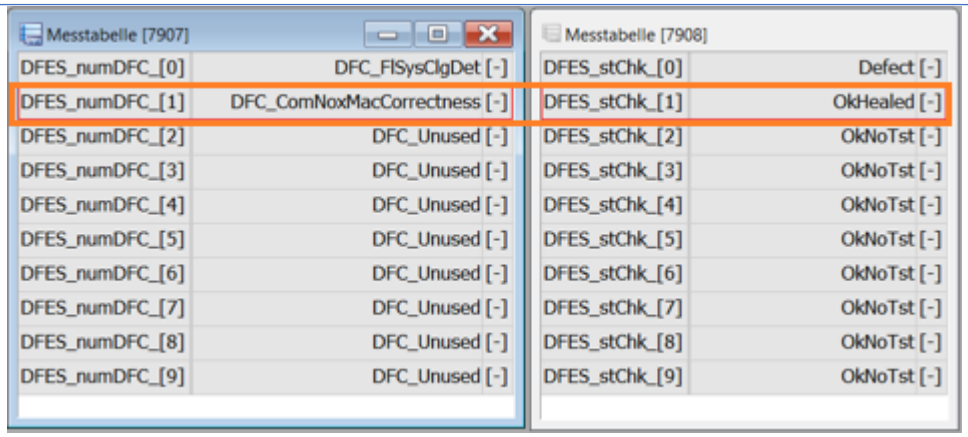


Figure 19 The error flag of MAC tampering is not set when only double entries are sent.

Test case 7: the Raspberry Pi acts as a foreign node and sniffs the communication. If the random number frame is received, the data in the frame is tampered and injected back to the CAN bus communication channel. The results show that this tampered random number is also used by the sensor control unit to calculate the MAC. With this wrong MAC the authentication fails on the ECU side.

Test case 8: the received MAC on the Raspberry Pi node is tampered and injected back to the communication between ECU and SCU. The test results show that the ECU does not ignore this tampered MAC, set the error flag and treats the communication as unauthorized.

Test case 9: on the Raspberry Pi node the received sensor data is tampered and injected back to the communication. This tampered data is used by ECU to calculate the MAC, but the MAC sent by SCU does not use the tampered data. Due to this reason the authentication is failed, as the MAC which is calculated by ECU is not the same as the MAC sent by sensor. The injected tampered data invalidated the integrity of transmitted data. The result shows that the ECU can detect the tampering and set the error flag.

Test case 10: a special test to check the behaviour of the software if the CAN bus frame with random CAN-ID values is injected to the CAN bus communication. During the test it can be seen that the vehicle reports immediately a lot of errors if the random message has high priority (Figure 20).

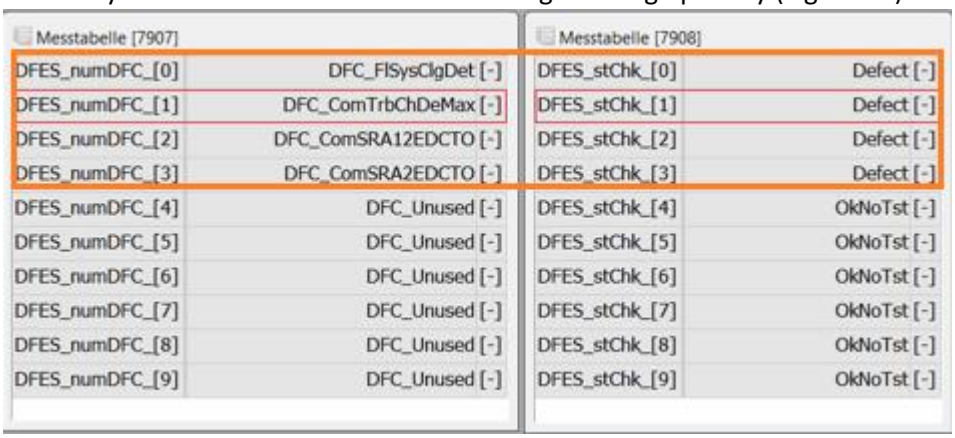


Figure 20 Error flags when injecting random CAN-ID

5.3.3.3 Conclusion

Every frame injection which contains tampered data can be detected by the ECU. The implemented measure of SecOC light against failed communication authentication and data transmission integrity is sufficient.

5.3.4 Replay attack

A replay attack (also known as a repeat attack or playback attack) is a form of network attack in which valid data transmission is maliciously or fraudulently repeated or delayed (Ref.). This is carried out either by the originator or by an adversary who intercepts the data and re-transmits it. The goal of this test is to validate the authentication requirement of the communication between ECU and SCU.

5.3.4.1 Information gathering

The setup of this test is the same as man-in-the-middle attack in subsection 5.3.1.

5.3.4.2 Test execution

Test case 11: The code in the Raspberry Pi logs some amount of data which is sent from SCU to ECU. Then it drops all data from SCU to ECU. Every time a random number frame from ECU is received, it replays the logged data to the communication. In this way, the test code tries to fake the authentication process. But because the MAC is calculated with the random number from ECU and the sensor data, the action to replay every time the same MAC to the CAN bus communication should be detected by the ECU as unauthorized data transmission.

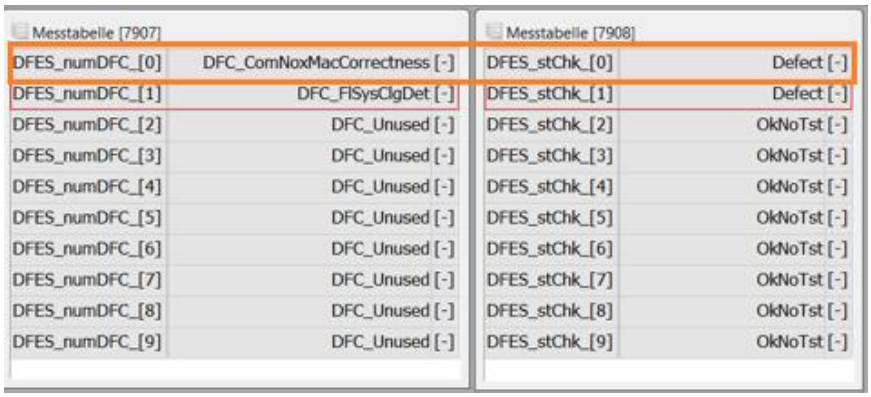


Figure 21 Error flags of replay attack

The parameter checks with INCA as shown in Figure 21 also shows that the error flag is set by ECU and indicates the unauthorized communication.

5.3.4.3 Conclusion

The tampering using replay attack was detected by the ECU. The secure mechanism against unauthenticated data transmission and data integrity attack is sufficient.

5.4 Analogue signal tampering of NOx Sensor

During the Hackathon event in DIAS T3.4, the analogue signal tampering of the NOx Sensor was proposed as a new potential attack path. Although the digital signal sent from SCU to ECU is protected by SecOC Light, there is no protection on the analogue signal of the NOx sensor. The data integrity can be damaged if the analogue signal is tampered. The SecOC Light cannot protect the analogue signal. For this reason, an after-event analysis was made.

5.4.1 Modifying the analogue signal of NOx sensor

5.4.1.1 Information gathering

The NOx sensor contains two main components, the physical sensor and the SCU. They are connected with each other by 6 wires. The principle of NOx sensor is shown in Figure 22. The physical component of the NOx Sensor has two main cells. The first cell receives the exhaust gases and pumps the oxygen in the exhaust gases, so it does not interfere with the NOx measurement in the second cell. The

remaining gases diffuse into the second cell where a reducing catalyst causes NOx to decompose into N2 and O2. As with the first cell, a bias of -400 mV applied to the electrode dissociates the resulting O2 which is then pumped out of the cell; the pumping current of the second cell is proportional to the amount of oxygen from the NOx decomposition.

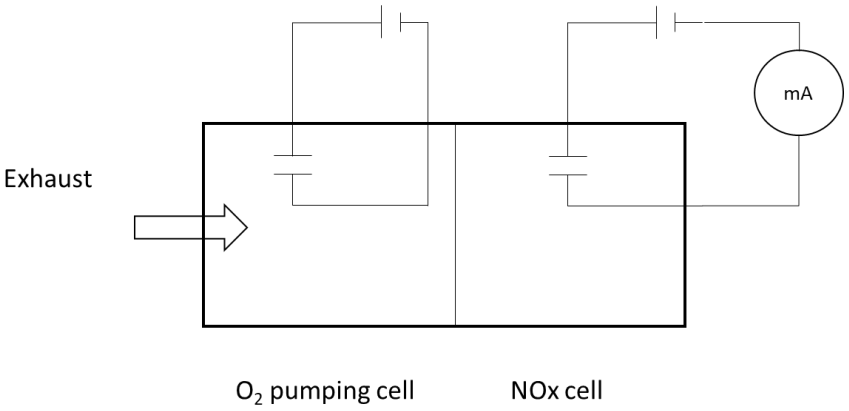


Figure 22 Principal diagram of NOx sensor

The pumping current of the second cell is then evaluated and sent by SCU to ECU.

5.4.1.2 Test execution

Since this attack was not analysed in the TARA of D4.1 [125], a new TARA is performed. At the same time, a test of tampering the analogue signal has also been implemented.

TARA method: to be consistent with the TARA approach defined in D4.1, the same method based on SAE J3061 is applied. The D4.1 was done at the first phase of the DIAS project and at that time the latest standard ISO 21434 was not published. Therefore, the TARA methodology used in DIAS is based on SAE J3061 and some adaptations have been implemented considering the specific requirements in the DIAS project. These adaptations are mainly introducing one impact parameter, the environmental impact and one threat parameter, financial motive [125].

- Environmental impact: it replaced the standard impact parameter, the operational impact, considering that the operational impact is not of higher significance in the context of DIAS project. Instead, the environmental impact of the attack is directly relevant to the Environmental Protection System (EPS), which is the focus of the DIAS project.
- Financial motive: it is included as additional parameter to calculate the threat level because it is highly likely a motive for an attack in the DIAS project.

TARA results:

- Attack: attacker emulates the analogue signals of NOx sensor.
- Threat: deactivation of Diesel Exhaust Fluid (DEF) dosing resulting from loss of integrity of the analogue signals of the upstream and downstream NOx sensor.

The evaluation of the threat level and its justification is summarised in Table 12.

Table 12 Threat level of analogue signal tampering

Financial motive	Expertise	Knowledge of the target	Window of opportunity	Equipment required	Sum of threat	Threat level

Low (2)	Multiple Expert (3)	Sensitive (2)	Large (1)	Multi bespoke (3)	11	None (0)

Justification:

Financial Motive: The cost of AdBlue consumption which could be saved is estimated up to 725 Euro per year (The SCR efficiency is reduced by approx. 25% considering the OBD limit, therefore up to 25% saving of the AdBlue cost per year). Considering this attack cannot be easily replicated to other vehicles, the financial motive is rated as Low.

Expertise: Expert level for diagnostic and analogue signal knowledge is required.

Knowledge of the Target: Sensitive information are required.

Window of Opportunity: Manipulations of analogue part (wiring) are detectable in PTI (Periodic Technical Inspections) and roadside. The Window of opportunity is large, but not unlimited.

Equipment Required: Special devices and customized equipment and tools are required. It is not easy for the attacker to obtain, therefore the equipment is rated as Multi bespoke (multiple special devices are required).

In Table 13 the impact level and its justification are summarised.

Table 13 Impact level of analogue signal tampering

Financial	Environmental	Privacy	Safety	Sum of impact	Impact level
Low (10)	Low (10)	No impact (0)	No impact (0)	20	Medium (2)

Justification:

Financial: if the tampering is successful, the OEM/supplier will have a loss of reputation. However, a significant loss is not expected.

Environmental: the reduced AdBlue consumption has to be under the plausibility limit in order to not cause fault codes. The environmental impact is rated low.

Privacy: There is no impact on privacy.

Safety: There is no impact on safety.

The above evaluated impact level and threat level are combined together to obtain the security level for this attack path, Quality Management (QM). Above this the plausibility check on the ECU protects against the tampering of the analogue signal of NOx sensor. The risk is accepted.

Test case 12: as described already in the previous section, the analogue signal of the NOx sensor is a current signal in nano ampere range. It is very difficult to measure it with a normal device. To avoid the use of special measurement devices and inaccurate measurements, it is decided to execute the test in an environment which contains the constant concentration of NOx gas. The sensor is mounted in pipe with constant concentration of sample gas as shown in Figure 23. The CAN bus signal is logged as indirect measurement of the sensor data. An inline resistor is added on the connection wires between the sensor and the SCU to tamper the electronic behaviour of the sensor. The measurement is compared with the measurement without any modification of the electronic of the sensor. So if it is successful, the CAN bus log should reflect the change of the analogue signal.

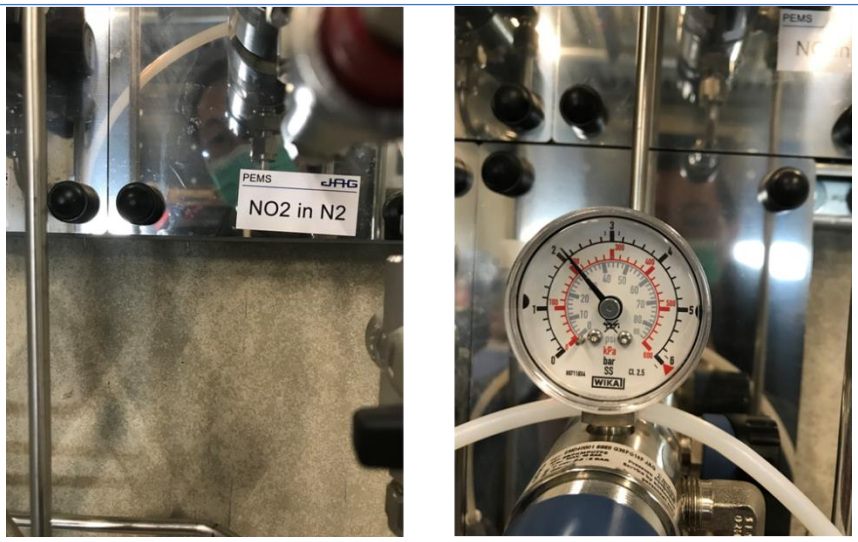


Figure 23 Sample gas

It can be seen from the result that the analogue signal could not be changed with this approach. Further analysis of the high level circuit shows that the signal is only dependent on the concentration of NOx in the near of NOx pump electronic. It needs to be pointed out that the test results facilitate the evaluation of threat level in the TARA analysis by rating the expertise and equipment of the attack more difficult. As illustrated in Figure 5 of Section 3.2.3, the mutual benefit between testing and risk assessment has been proven.

5.4.1.3 Conclusion

To modify the physical sensor and the SCU circuit, special and different expertise are needed. The current of the NOx signal is in nano-ampere range which makes the modification also quite difficult. The modification cannot be hidden easily from the road check. In addition, the ECU has also plausibility check functionality to detect abnormal signals. This makes the modification of the analogue signal with low reward and high effort. The risk of this type of tampering is extremely low.

5.5 Unauthorized flash memory access of SCU

In the Hackathon event of T3.4 another attack path, the unauthorized memory access of SCU through hardware pins, is documented as a new potential attack path. If the chip contains the pins, through which the memory can be dumped. with a successful attack, the software of SCU can be tampered to deliver tampered digital signal to ECU. Due to the hardware limitation, there is no HSM on SCU, which makes theoretically the unauthorized dump of SCU memory possible.

5.5.1 Unauthorized memory access through hardware pins

5.5.1.1 Information gathering

In the scope of this test, the hardware example which is available, is a Bosch NOx sensor with SCU. The complete circuit board of the SCU is sealed with plastic. Once the plastic is removed, it cannot be returned to the original state. Only necessary pins are remained as interfaces. If the software can be modified successfully, the tampered SCU can send the fake sensor data to ECU without being detected by the SecOC Light.

5.5.1.2 Test execution

Similar to the analogue signal tampering attack described in subsection 5.4.1.2, TARA is generated for this new attack path using the same approach.

TARA results:

- Attack: attacker flashes the NOx sensor microcontroller in the chip to report reduced AdBlue consumption values.
- Threat: reduced AdBlue consumption and thereby increased NOx emission resulting from loss of integrity of the software on the NOx SCU.

The evaluation of the threat level and its justification is summarised in Table 14.

Table 14 Threat level of SCU memory dump attack

Financial Motive	Expertise	Knowledge of the Target	Window of Opportunity	Equipment Required	Sum of threat	Threat level
Low (2)	Multiple Expert (3)	Sensitive (2)	Unlimited (0)	Specialized (1)	8	Low (1)
Justification: Financial Motive: The cost of AdBlue consumption, which could be saved, is estimated up to 725 Euro per year (The SCR efficiency is reduced by approx. 25% considering the OBD limit, therefore up to 25% saving of the AdBlue cost per year). Considering that this attack cannot be easily replicated to other vehicles, the SCU might be broken during opening, the financial motive is rated as Low. Expertise: Multiple experts are needed, such as hardware, SCU software, Diagnostics. Knowledge of the Target: Sensitive information are required. The flashing procedure and memory are sensitive information. Window of Opportunity: If the attacker has access to the vehicle, he/she will have the access to the SCU all the time. Equipment Required: Special connectors are needed, e.g., SOIC (Small Outline Integrated Circuit)-test clips [143]						

The impact level is summarised in Table 15.

Table 15 Impact level of SCU memory dump attack

Financial	Environmental	Privacy	Safety	Sum of impact	Impact level
Low (10)	Low (10)	No impact (0)	No impact (0)	20	Medium (2)
Justification: Financial: If the tampering is successful, the OEM/supplier will have a loss of reputation. However, a significant loss is not expected. Environmental: The reduced AdBlue consumption has to be under the plausibility limit in order to not cause fault codes. The environmental impact is rated low. Privacy: There is no impact on privacy. Safety: There is no impact on safety.					

Combing the above impact level rating and the threat level rating, the security level of this attack path is rated as Low. Above this the ECU protects against the tampering of the NOx SCU data. Hence the risk is accepted.

Test case 13: in parallel of TARA creation, it was tried to remove the plastic out of the circuit board of SCU with mechanical tools. As shown in Figure 24, after removing the plastic on one side, there is no space left between the board and the plastic case during manufacturing. To remove the complete plastic case with mechanical tools to access the pins of the chip without damage to the electronic components on the board (which could be easily detected) is a significant challenge. To remove the plastic with heat has also the high risk to damage the electronic components and the content of memory, which could also be easily detected.

Figure 24 SCU with removed plastic

5.5.1.3 Conclusion

The reward and effort ratio of this type of attack is low. With the facilitation of the test results, the TARA is adjusted accordingly. The final risk level is evaluated as low.

5.6 Summary of security testing results

In the following table, the security testing results described in this chapter are summarised.

Table 16 Summary of security testing results

Test category	Test case	Description	Test results
Brute force attack (Subsection 5.2.1)	1	Repeatedly sending a random signature guess without requesting a new challenge during ECU reprogramming	The attack failed. The test passed.
	2	Repeatedly sending a random signature guess with requesting a new challenge during ECU reprogramming	The attack failed. The test passed.
Tampering attack (Subsection 5.2.2)	3	Modifying part of the software and dataset to be flashed on ECU during ECU reprogramming	Tampering is detected. The test passed.

MITM attack (Subsection 5.3.1 and 5.3.2)	4	Redirecting the frames from sensor to attacker and tamper the data in the received frame and send the tampered data to ECU	Tampering is detected. The test passed.
	5	Redirecting the frames from Sensor to attacker and tamper the MAC in the received frame and send the tampered data to ECU	Tampering is detected. The test passed.
Fault frame injection attack (Subsection 5.3.3)	6	Sniffing the CAN, injecting the random number frame and MAC frame back	No impact as far as no tampering in the injection (robust against double MAC message). The test passed.
	7	Sniffing the CAN, tampering and inject the random number frame back	Tampering is detected. The test passed.
	8	Sniffing the CAN, tampering and inject the MAC frame back	Tampering is detected. The test passed.
	9	Sniffing the CAN, if data frame from sensor received then tampering the data and inject it back	Tampering is detected. The test passed.
	10	Injecting frames with unknown arbitration ID	Tampering is detected. The test passed.
Replay attack (Subsection 5.3.4)	11	Logging the frames between random number from ECU and MAC from SCU and replaying the logged data(pre-log) for the next receiving of random number	Tampering is detected. The test passed.
Analog signal of NOx sensor tampering (Subsection 5.4.1)	12	Adding electronic elements and cutting wires	The analogue signal cannot be tampered with simple modification. Combined with plausibility check on ECU, the risk of this attack is very low.
Unauthorized SCU flashing in the chip (Subsection 5.4.2)	13	Removing the case and accessing the NOx SCU through hardware pins	The sealed case cannot be removed easily without damaging the chips. Reward-effort ratio is too low.

6 Concept review of the Key Exchange RSA Asymmetric Approach

6.1 Scope of the concept review

The objective of this chapter is to present the findings of a concept review performed by the DIAS partners on the Key Exchange RSA Asymmetric Approach developed in the DIAS project. A potential approach would be to follow the TARA methodology applied in DIAS D4.1 [125] which is based on SAE J3061 [138]. The focus of the concept review is on the threat assessment because the remediation aspects will not be implemented anyway in time for the end of the project. Considering that the project is on tampering, the threat concept has a wider meaning than just cybersecurity and it includes tampering attacks. On the other side, the concept review also discusses some deployment aspects.

6.2 Description of the Key Exchange RSA Asymmetric Approach

One of the drivers of that approach is to establish a process of key exchange suitable for sensor control units (SCU) where the performance of microcontroller on runtime and resources are limited. In this approach the main goal is to distribute a unique key in between two communication partners (e.g., SCU and ECU) used for a secure communication (e.g., on CAN).

The overall schema is presented in Figure 25.

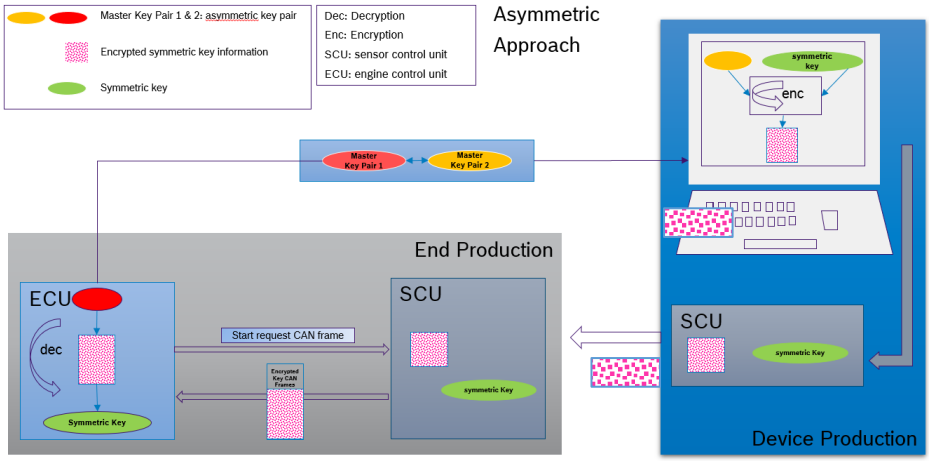
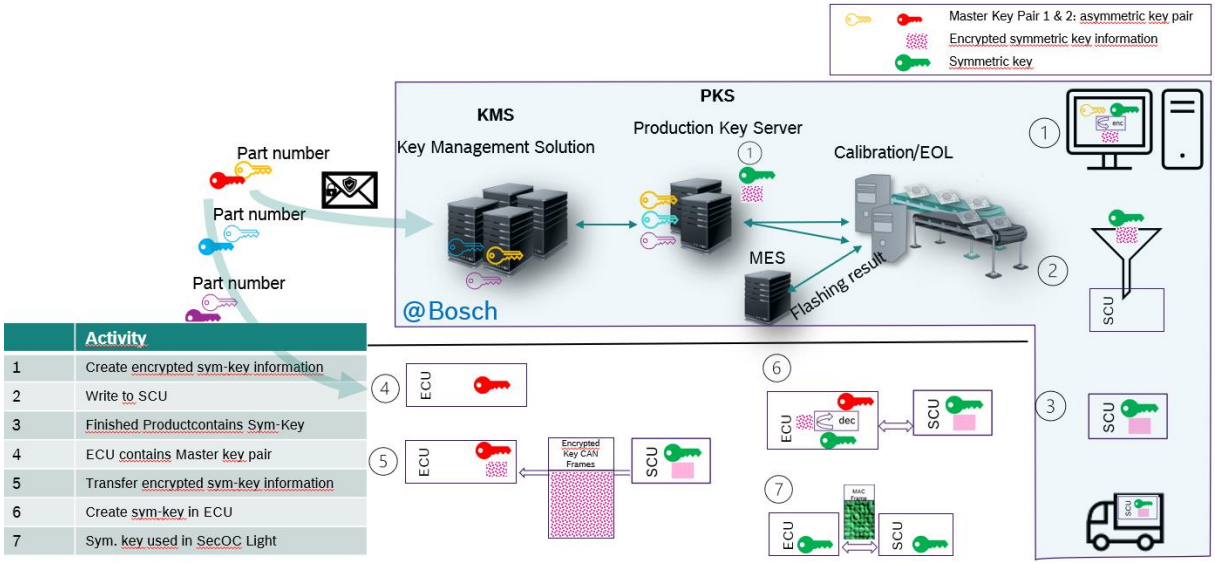


Figure 25 Key Exchange RSA Asymmetric Approach Concept

In the proposed approach, the car producer will generate a Master Key Pair #1 & #2. This key pair is an asymmetric key pair where only the Key Pair #2 will be provided to the producer of the SCU. The device production will use the Master Key Pair #2 to generate out of a unique and random generated symmetric key as an encrypted package. In the next step those generated information for each part will be bootstrapped in the SCU during the production phase. The SCU will be manufactured as in normal production process and shipped to the car producer. During the End Production phase, the SCU manufacturer will be requested to send the encrypted package to the ECU which holds the Master Key Pair #1. The ECU must be bootstrapped with this information. With this information, the ECU can decrypt the encrypted package and gets the symmetric key, which is used as common information between the ECU and SCU for e.g., secure communication. Alternatively, the Tester or a Workshop, (e.g., for Flashing, Calibration etc.) communicating with the ECU in End Production is holding / getting the info about Master Key pair #1 and can decrypt the received encrypted package to get the symmetric key for bootstrapping inside the ECU.

The overview in Figure 26 shows more in detail the process-steps, their necessary environments, and work-steps.



	Activity
1	Create encrypted sym-key information
2	Write to SCU
3	Finished Productcontains Sym-Key
4	ECU contains Master key pair
5	Transfer encrypted sym-key information
6	Create sym-key in ECU
7	Sym. key used in SecOC Light

Figure 26 Key Exchange RSA Asymmetric Approach Production Concept

Beginning from left, the car producer generates a Master Key Pair #1 and #2. The Master key Pair #1 is a private Key and #2 is the public Key. The Master Key Pair #2 is provided in a secured way (i.e., because it is asymmetric cryptographic key material) to the supplier by encrypted mail. If the master key has to be subsequently replaced during a migration effort or due to the obsolescence of the underlying cryptographic algorithm (e.g., because a NIST analysis has shown that the algorithm cannot be considered secure any longer), the vehicle manufacturer and the supplier of the SCU has to jointly define the logistic workflow (i.e., distribution channel and protocol) for changing the Master Key Pair #2.

Note that in Figure 26, the Master Key Pair #2 and in respect to it the Master Key Pair #1 is changing for each Part number. Alternatively, it can be decided (to improve the overall robustness of the framework) to change a Key Pair often and to use it for the whole vehicle by the vehicle manufacturer.

The supplier takes the key material into the KMS (Key Management Solution) environment. KMS is a server environment which is centralized to be able to share confidential information to server environments directly in the production lines. The Production Key Server (PKS) retrieves the key material provided by the vehicle manufacturer and the related meta data to correlate the key material to the product information. If there is a need of producing a Sensor Control Unit (SCU), the PKS produces a random symmetric key which is unique for each part. PKS will use the standard NIST SP800-108 [133] with HMAC-SHA256 in counter mode as PRF (pseudo random function). Furthermore, it creates the encrypted package with the help of Master Key pair #2. Therefore standard NIST SP800-56BR1 [134] PKCS#1 v2.2/2.1 RSAES-OAEP is used with Padding and Random Data to produce the encrypted package. This information will be written to the SCU in the Calibration/EOL station placed at the end of the production line of the SCU. The information about each part will be stored in MES (Manufacturing Execution System). During the production phase of the vehicle, the ECU contains the Master Key Pair and request the SCU for sending the encrypted packages on e.g., CAN-bus in-vehicle network. The ECU decrypts the encrypted package on the CAN-bus with the help of Master Key Pair #1 (also called private key in respect to asymmetric key material) to get the data of symmetric key and store it in its secure memory. Now both communication partners (ECU and SCU) are assigned to same secret (e.g., symmetric key) and established an authenticated communication.

To ensure the protection of the key derivation function in the KMS and PKS, the internal security directives for the server environment and infrastructure are based on the Standard ISO 27001 – 2013 [135].

6.3 Threat and vulnerability analysis

6.3.1 List of attack vectors

This subsection (inspired by TARA) is focused on the definition of the potential attack vectors on the different elements of the proposed architecture as shown in Figure 26. During concept review session held between DIAS partners different potential attack vectors are discussed and the following attack vectors listed in Table 17 are identified as valid attacks.

Table 17 List of attack vectors

ID	Attack	Explanations
A001	The attacker steals the asymmetric key pair (might be outside of the organization) and generates symmetric key in SCU and ECU.	The asymmetric key pair were generated by car producer and will be distributed to the producer of small control units. There might be different suppliers of the small control units, thus different companies will have the key information. The key can be stolen and there is no traceability where it was lost.
A002	Attacker accesses the chip of SCU to retrieve the cryptography key.	The symmetric key on SCU is not saved in secured memory (HSM) and the key is static in the SCU lifetime. If the key is disclosed, the attacker can tamper the data sending from SCU to ECU.

6.3.2 Risk and vulnerability analysis method and results

Based on the attack vectors identified in the previous subsection, a risk analysis is performed to verify a) how plausible is the attack (what are the needed capabilities or the motivations of an attacker) – threat level; b) what is the potential impact (damage of a critical asset) – impact level. The TARA method defined in D4.1 is applied to this concept review in order to be consistent with the TARA done in earlier stage of the DIAS project. The description of the TARA method can be found in D4.1 [125] and it is also shortly introduced in Subsection 5.4.1.2 of this deliverable. Below the two identified attack vectors are analysed, the threat level and impact level are evaluated respectively, and the final security level are derived from the evaluation.

A001:

- Attack: The attacker steals the asymmetric key pair (might be outside of the organization) and generates symmetric key in SCU and ECU.
- Threat: Fake authentication on the ECU-SCU SecOC light resulting from loss of confidentiality of the asymmetric key pair.
- Threat level: the rating of the threat level and the corresponding justifications are listed in Table 18.

Table 18 Threat level of attack A001

Financial Motive	Expertise	Knowledge of the Target	Window of Opportunity	Equipment Required	Sum of threat	Threat level
Medium (1)	Multiple Expert (3)	Critical (3)	Small (3)	Bespoke (2)	11	None (0)
Justification: Financial Motive: The cost of AdBlue consumption, which could be saved, is estimated up to 725 Euro per year (The SCR efficiency is reduced by approx. 25% considering the OBD limit, therefore up to 25% saving of the AdBlue cost per year). Considering the asymmetric key pair is unique for each part number, not each part, this attack can be replicated to other vehicles, the financial motive is rated as medium. Expertise: Multiple experts are needed, such as cryptography, microcontroller, NOx value, IT structure. Knowledge of the Target: The cryptography is critical information. Window of Opportunity: It is not easy to access the key server. Equipment Required: Computer, CAN communication tools, tools can be purchased. special scripts need to be developed.						

- Impact level: the rating of the impact level and the corresponding justifications are listed in Table 19.

Table 19 Impact level of attack A001

Financial	Environmental	Privacy	Safety	Sum of impact	Impact level
Low (10)	Low (10)	No impact (0)	No impact (0)	20	Medium (2)
Justification: Financial: it is not clear where the asymmetric key pair is disclosed, it will result in less reputation loss and consequently low financial loss. The IT structure follows the secure standard, if there is no deviation from the standard, there is no reputation loss. The financial loss is rated as low. Environmental: The reduced AdBlue consumption has to be under the plausibility limit in order to not cause fault codes. The environmental impact is rated low. Privacy: There is no impact on privacy. Safety: There is no impact on safety.					

Security level: as the impact level is medium and the threat level is none, the resultant security level is rated as QM.

A002:

- Attack: Attacker accesses the chip of SCU to retrieve the cryptography key.
- Threat: Fake authentication on the ECU-SCU SecOC light resulting from loss of confidentiality of the symmetric key.
- Threat level: the rating of the threat level and the corresponding justifications are listed in Table 20.

Table 20 Threat level of attack A002

Financial Motive	Expertise	Knowledge of the Target	Window of Opportunity	Equipment Required	Sum of threat	Threat level
Low (2)	Multiple Expert (3)	Sensitive (2)	Unlimited (0)	Specialized (1)	8	Low (1)
Justification: Financial Motive: The cost of AdBlue consumption which could be saved is estimated up to 725 Euro per year (The SCR efficiency is reduced by approx. 25% considering the OBD limit, therefore up to 25% saving of the AdBlue cost per year). Considering the cryptography key for each vehicle is unique, and also this attack cannot be easily replicated to other vehicles, the SCU might be broken during opening, the financial motive is rated as Low. Expertise: Multiple experts are needed, such as hardware, microcontroller memory. Knowledge of the Target: Microcontroller memory is sensitive information. Window of Opportunity: If the attacker has access to the vehicle, he will have the access to the SCU all the time. Equipment Required: Special connectors are needed (SOIC-test clips) [143].						

- Impact level: the rating of the impact level and the corresponding justifications are listed in Table 21.

Table 21 Impact level of attack A002

Financial	Environmental	Privacy	Safety	Sum of impact	Impact level
Low (10)	Low (10)	No impact (0)	No impact (0)	20	Medium (2)
Justification: Financial: If the tampering is successful, the OEM/supplier will have a loss of reputation. However, a significant loss is not expected. Environmental: The reduced Adblue consumption has to be under the plausibility limit in order to not cause fault codes. The environmental impact is rated low. Privacy: There is no impact on privacy. Safety: There is no impact on safety.					

Security level: as the impact level is medium and the threat level is low, the resultant security level is rated as low.

6.4 Deployment aspects

This section discusses the deployment aspects including the potential complexity or scalability of the implementation and deployment of the approach in the real world. In particular, it can be discussed what type of infrastructure should be put up or which interfaces with existing infrastructures should be set up.

The following aspects are highlighted:

- The cryptographic system and algorithms cannot remain the same forever and a migration process should be defined to ensure that the existing cryptographic materials and pairing is smoothly replaced. One possibility would be the physical replacement of the ECU and SCU but this would require massive recalls. Another solution would be to leave the existing vehicles with the already installed cryptographic material even if the corresponding algorithm is considered obsolete and not secure any longer. Another possibility would be to provide a migration mechanism with channels to replace the cryptographic material even if such design solution could be complex to implement.
- In the deployment process, the specific roles of the different stakeholders should be clearly defined. In particular, it should be defined what type of access rights each role has, to which interface and protocols (e.g., transmission of cryptographic material) the roles are related and if there are dependencies among them. A possible list of roles could: vehicle manufacturers, ECU supplier, SCU supplier, manufacturer workshop, third party workshop, law enforcer (as the framework will be implemented to fulfil a regulation), the vehicle owner and so on.
- The previous sections do already define a list of potential standards, which can support the provided framework. It may also be useful to identify potential tools, which the workshop or manufacturer must equip.
- The supply chain between the SCU, ECU, workshop and vehicle manufacturer should be protected to ensure that the cryptographic material is not tampered or replaced.

6.5 Summary of the analysis

A concept review was performed on the security framework proposed by DIAS partners for the definition, derivation and distribution of cryptographic materials need to ensure the secure in-vehicle connectivity. A list of attack paths was identified, and the security level was analysed using TARA approach. Overall, the proposed security framework seems resilient to the identified threats. Some aspects related to deployment and operations are not fully described (but it was understood that it is not in the scope of the DIAS project) in the concept, and they should be addressed in a market deployment of the security framework.

7 Conclusions

This deliverable fulfilled three main tasks: to provide an overview of the validation and verification approaches (including risks assessment), which could be applied to the vehicle tampering domain, to report on the validation and verification testing activities of the DIAS demonstrator and to report on the concept review of the DIAS solutions developed during the DIAS project.

Each of the three tasks is instrumental to support future actions both at the regulatory and standardization level to identify risks for tampering and related countermeasures. This is a new field where the increasing digitalization of the automotive sector in terms of connectivity (e.g., with 3GPP [122]), computing capabilities and networks generate new significant opportunities but also new threats. In particular, tampering with regulation has often strong drivers for the malicious/tampering party in terms of economic gains, illegitimate increase of performance or creating competitiveness bias (the tamperer has an economic advantage). The considerable amount of work and methodologies in risk assessment and cybersecurity can be employed in the detection of the tampering risks and definition of related countermeasures taking in consideration the differences between tampering and cybersecurity as tampering is focused on specific goals and objectives.

Considerable effort (beyond the scope of DIAS) must still be invested in the deployment and operational aspects of the solutions proposed in DIAS. This is future work, which should be done in synergy with on-going regulatory and standardization activities.

8 References

[1]. European Union Agency for Network and Information Security (ENISA), "Cyber security and resilience of smart cars. Good practice and recommendations," p.84, 2017. [Online]. Available: https://publications.europa.eu/en/publication-detail/-/publication/13d4bf8d-e9de-11e6-ad7c-01aa75ed71a1

[2]. Society of Automotive Engineers (SAE), "J3061 - Cybersecurity Guidebook for Cyber-Physical Vehicle Systems," p. 128, 2016. [Online]. Available: http://standards.sae.org/wip/j3061/

[3]. A. Lekidis and I. Barosan, "Model-based simulation and threat analysis of in-vehicle networks," in IEEE International Workshop on Factory Communication Systems - Proceedings, WFCS, vol. 2019-May. Institute of Electrical and Electronics Engineers Inc., 5.2019, p. 8757968. [Online]. Available: https://research.tue.nl/en/publications/model-based-simulation-and-threat-analysis-of-in-vehicle-networks

[4]. Robert Bosch GmbH, Automotive Electrics and Automotive Electronics: Systems and Components, Networking and Hybrid Drive, R. B. GmbH, Ed. Springer, 2007. [Online]. Available: https://www.amazon.es/Bosch-Automotive-Electrics-Electronics-Professional-ebook/dp/B00H6BNWI8

[5]. V. H. Le, J. den Hartog, and N. Zannone, "Security and privacy for innovative automotive applications: A survey," pp. 17–41, 11.2018. [Online]. Available: https://www.sciencedirect.com/science/article/abs/pii/S014036641731174X

[6]. C. Smith, The Car Hacking Handbook: A Guide for the Penetration Tester, 2016.

[7]. G. De La Torre, P. Rad, and K.-K. R. Choo, "Driverless vehicle security: Challenges and future research opportunities," Future Generation Computer Systems, vol. 108, pp. 1092–1111, 7 2020. [Online]. Available: https://linkinghub.elsevier.com/retrieve/pii/S0167739X17315066

[8]. J. Petit and S. E. Shladover, "Potential Cyberattacks on Automated Vehicles," IEEE Transactions on Intelligent Transportation Systems, pp. 1–11, 2014. [Online]. Available: https://ieeexplore.ieee.org /document/6899663

[9]. M. Hashem Eiza and Q. Ni, "Driving with Sharks: Rethinking Connected Vehicles with Vehicle Cybersecurity," IEEE Vehicular Technology Magazine, vol. 12, no. 2, pp. 45–51, 6 2017. [Online]. Available: https://ieeexplore.ieee.org/document/7908939/

[10]. Allot, "Connected Cars Attack Vulnerabilities," 2018. [Online]. Available: https://www.allot.com/resources/TB_CONNECTED_CARS.pdf

[11]. UNECE World Forum for Harmonization of Vehicle Regulations, "Proposal for a new UN Regulation on uniform provisions concerning the approval of vehicles with regards to cyber security and cyber security management system," p. 27, 2020. [Online]. Available: http://www.unece.org/fileadmin/DAM/trans/doc/2020/wp29grva/ECE-TRANS-WP29-2020-079-Revised.pdf

[12]. European Union Agency for Network and Information Security, "ENISA good practices for security of Smart Cars," p. 132, 2019. [Online]. Available: https://www.enisa.europa.eu/publications/smart-cars

[13]. European Union Agency for Network and Information Security (ENISA), "Cyber Security and Resilience of smart cars," 2017. [Online]. Available: https://www.enisa.europa.eu/publications/cyber-security-and-resilience-of-smart-cars

[14]. UNECE, "UN Regulation on uniform provisions concerning the approval of vehicles with regards to software update and software updates management system," pp. 1–15, 2020. [Online]. Available: https://undocs.org/en/ECE/TRANS/WP.29/GRVA/2020/4

[15]. CNSSI, "CNSSI No. 4009: Committee on National Security Systems (CNSS) Glossary," 2015.

[16]. MITRE, Common Weakness Scoring (CWE) System, 2011. [Online]. Available: https://cwe.mitre.org/cwss/cwss_v1.0.1.html
[17]. J. R. C. Nurse, S. Creese, and D. D. Roure, "Security Risk Assessment in Internet of Things Systems," IEEE Computer Society, IT Pro, 2017.
[18]. MITRE, "Common Weakness Risk Analysis Framework (CWRAF)." [Online]. Available: https://cwe.mitre.org/cwraf/
[19]. FIRST, Common Vulnerabilities Scoring System (CVSS), 2014. [Online]. Available: https://www.first.org/cvss.
[20]. R. a. R. a. C. Caralli, J. F. Stevens, L. R. Young, and W. R. Wilson, "Introducing OCTAVE Allegro: Improving the Information Security Risk Assessment Process," US Dept of the Air Force, Tech. Rep. May 5, 2007. [Online]. Available: http://www.dtic.mil/docs/citations/ADA470450
[21]. C. J. Alberts, A. J. Dorofee, J. F. Stevens, and C. Woody, "OCTAVE-S Implementation Guide, Version 1," Tech. Rep., 2005. [Online]. Available: https://resources.sei.cmu.edu/asset_files /Handbook/2005_002_001_14273.pdf
[22]. Microsoft, DREAD scheme. [Online]. Available: https://msdn.microsoft.com/en-us/library /ff648644.aspx
[23]. Openstack, Security/OSSA-Metrics. [Online]. Available: https://wiki. openstack.org/wiki/Security /OSSA-Metrics#Calibration
[24]. A. B. Garcia, R. F. Babiceanu, and R. Seker, "Trustworthiness Requirements and Models for Aviation and Aerospace Systems," in 2018 Integrated Communications, Navigation, Surveillance Conference (ICNS). Herndon, VA: IEEE, 2018, pp. 1–16.
[25]. NCCgroup, "Threat Prioritisation: DREAD Is Dead, Baby?" https://www.nccgroup.trust/uk /about-us/newsroom-and-events/blogs/2016/march/threat-prioritisation-dread-is-dead-baby/, 2016.
[26]. OWASP, OWASP Application Security Verification Standard (ASVS) Project. [Online]. Available: https://www.owasp.org/index.php/OWASP_Risk_Rating_Methodology
[27]. A. Söderberg-rivkin, "Threat Modelling and Risk Assessment," Tech. Rep. August, 2014. [Online]. Available: https://odr.chalmers.se/handle/20.500.12380/202917
[28]. VERACODE, "Veracode." [Online]. Available: https://help.veracode.com/r/review_main
[29]. Cenzic, "HARM Score: Approaches to Quantitative Risk Analysis for Web Applications," in Bay Area OWASP event, 2010. [Online]. Available: https://es.slideshare.net/Cenzic/ harm-score-approaches-to-quantitative-risk-analysis-for-web-applications
[30]. CCRA, "Common Criteria, Assurance Continuity, CCRA Requirements. Version 2.1," 2012.
[31]. "Arrangement on the Recognition of Common Criteria Certificates in the field of Information Technology Security," pp. 1–49, 2014. [Online]. Available: https://www.commoncriteriaportal .org/files/ CCRA-July2, 2014-RatifiedSeptember82014.pdf
[32]. C. Zhou and S. Ramacciotti, "Common Criteria: Its Limitations and Advice on Improvement," ISSA Journal, 2011. [Online]. Available: https://www.difesa.it/SMD_/Staff/Reparti/II/CeVa /Pubblicazioni/ Estere/Documents/CommonCriteria_ISSAJournal_0411.pdf
[33]. S. P. Kaluvuri, M. Bezzi, and Y. Roudier, "A Quantitative Analysis of Common Criteria Certification Practice," 2014, pp. 132–143. [Online]. Available: http://link.springer.com/10.1007/978-3-319-09770-1_12
[34]. CAR 2 CAR Communication Consortium, "Protection Profile V2X Hard- ware Security Module," 2018.
[35]. J. Eichler and D. Angermeier, "Modular Risk Assessment for the Development of Secure Automotive Systems," 31. VDI/VW-Gemeinschaftstagung Automotive Security, vol. VDI-Berich, no. January 2015, p. 11, [Online]. Available: https://www.researchgate.net/publication

/283087728_Modular_
risk_assessment_for_the_development_of_secure_automotive_systems
[36]. A. R. Ruddle, H. Mira, and S. Information, “Security requirements for automotive on-board networks based on dark-side scenarios. E-safety vehicle intrusion protected applications.” EVITA project, March 2009. [Online]. Available: https://www.researchgate.net/publication /46307752_Security_requirements_for_automotive-on-board_networks_based_on_dark-side_scenarios_Deliverable_D23_EVITA_E-safety_vehicle_intrusion_protected_applications
[37]. L. Aljoscha and M. Islam, “HEAling Vulnerabilities to ENhance Software Security and Safety - Project Proposal HAVENS,” 2016.
[38]. ETSI, “ETSI TS 102 165-1 Methods and protocols; Part 1: Method and pro forma for Threat, Vulnerability, Risk Analysis (TVRA),” 2017.
[39]. G. Macher, E. Armengaud, E. Brenner, and C. Kreiner, “A Review of Threat Analysis and Risk Assessment Methods in the Automotive Context,” 2016, pp. 130–141. [Online]. Available: http://link.springer.com/10.1007/978-3-319-45477-1_11
[40]. J. Hao and G. Han, “On the Modeling of Automotive Security: A Survey of Methods and Perspectives,” Future Internet, vol. 12, no. 11, p. 198, 11. 2020. [Online]. Available: https://www.mdpi.com/1999-5903/12/11/198
[41]. B. Sheehan, F. Murphy, M. Mullins, and C. Ryan, “Connected and autonomous vehicles: A cyber-risk classification framework,” Transportation Research Part A: Policy and Practice, vol. 124, pp. 523–536, 6 2019. [Online]. Available: https://linkinghub.elsevier.com/retrieve/pii /S096585641830555X
[42]. J. Cui and G. Sabaliauskaite, “On the Alignment of Safety and Security for Autonomous Vehicles,” no. November 2017, 2018. [Online]. Available: https://www.researchgate.net/publication /324006032_On_ the_Alignment_of_Safety_and_Security_for_Autonomous_Vehicles
[43]. H.-K. Kong, M. K. Hong, and T.-S. Kim, “Security risk assessment framework for smart car using the attack tree analysis,” Journal of Ambient Intelligence and Humanized Computing, vol. 9, no. 3, pp. 531–551, 6 2018. [Online]. Available: http://link.springer.com/10.1007/s12652-016-0442-8
[44]. J.-P. Monteuuis, A. Boudguiga, J. Zhang, H. Labiod, A. Servel, and P. Urien, “SARA: Security Automotive Risk Analysis Method,” in Proceedings of the 4th ACM Workshop on Cyber-Physical System Security. New York, NY, USA: ACM, 5 2018, pp. 3–14. [Online]. Available: https://dl.acm.org/doi/10.1145/3198458.3198465
[45]. C. Schmittner, Z. Ma, and P. Smith, “FMVEA for safety and security analysis of intelligent and cooperative vehicles,” Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), vol. 8696 LNCS, pp. 282–288, 2014. [Online]. Available: https://link.springer.com/chapter/10.1007/978-3-319-10557-4_31
[46]. C. Raspotnig, P. Karpati, and A. L. Opdahl, “Combined Assessment of Software Safety and Security Requirements,” Journal of Cases on Information Technology, vol. 20, no. 1, pp. 46–69, 1 2018. [Online]. Available: http://services.igi-global.com/resolvedoi/resolve.aspx?doi=10.4018 /JCIT.2018010104
[47]. M. Jouini and L. B. A. Rabai, “Threats Classification,” 2016, pp. 368–392. [Online]. Available: http://services.igi-global.com/resolvedoi/resolve.aspx? doi=10.4018/978-1-5225-0105-3.ch016
[48]. A. J. Neumann, N. Statland, and W. R.D, “Post-processing audit tools and techniques,” 1977.
[49]. K. A. da Costa, J. P. Papa, C. O. Lisboa, R. Munoz, and V. H. C. de Albuquerque, “Internet of Things: A survey on machine learning-based intrusion detection approaches,” Computer Networks, vol. 151, pp. 147–157, 3 2019. [Online]. Available: https://linkinghub.elsevier.com/retrieve/pii /S1389128618308739

[50]. J. Hughes and G. Cybenko, “Quantitative Metrics and Risk Assessment: The Three Tenets Model of Cybersecurity,” Technology Innovation Management Review, vol. 3, no. 8, pp. 15–24, 8 2013. [Online]. Available: http://timreview.ca/article/712

[51]. V. Aceituno, “Open Information Security Maturity Model.”

[52]. A. W. Rufi and Cisco Networking Academy Program., Network security 1 and 2 companion guide, 2007.

[53]. Web Application Security Consortium (WASC), “WASC threat classification,” 2010.

[54]. D. Hutchison, T. Kanade, J. Kittler, J. M. Kleinberg, F. Mattern, J. C, Mitchell, M. Naor, O. Nierstrasz, C. Pandu Rangan, B. Steffen, M. Sudan, D. Terzopoulos, D. Tygar, M. Y. Vardi, G. Weikum, M. Bartoletti, P. Degano, and G. L. Ferrari, “Security Issues in Service Composition,” in International Conference on Formal Methods for Open Object-Based Distributed Systems, vol. 4037. Springer, 2006, pp. 1–16. [Online]. Available: http://www.springerlink.com/index /10.1007/11768869_1

[55]. 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,” Reliability Engineering & System Safety, vol. 198, p. 106773, 6 2020. [Online]. Available: https://linkinghub.elsevier.com/retrieve/pii/S095183201830499X

[56]. G. Macher, R. Messnarz, E. Armengaud, A. Riel, E. Brenner, and C. Kreiner, “Integrated Safety and Security Development in the Automotive Domain,” 3 2017. [Online]. Available: https://www.sae.org/ content/2017-01-1661/

[57]. G. Macher, C. Schmittner, O. Veledar, and E. Brenner, “ISO/SAE DIS 21434 Automotive Cybersecurity Standard - In a Nutshell,” in Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), vol. 12235 LNCS. Springer Science and Business Media Deutschland GmbH, 2020, pp. 123–135.

[58]. S. Murdoch, M. Bond, and R. J. Anderson, “How Certification Systems Fail: Lessons from the Ware Report,” IEEE Security & Privacy Magazine, pp. 1–1, 2012. [Online]. Available: http://ieeexplore.ieee.org/document/6231616/

[59]. ENISA, “Good practices for security of Smart Cars,” 2019. [Online]. Available: https://www.enisa.europa.eu/publications/smart-cars

[60]. S. N. Matheu-Garcia, J. L. Hernandez-Ramos, A. F. Skarmeta, and G. Baldini, “Risk-Based Automated Assessment and Testing for the Cybersecurity Certification and Labelling of {{IoT}} Devices,” Computer Standards & Interfaces, vol. 62, pp. 64–83, 2019.

[61]. S. N. Matheu, J. L. Hernandez-Ramos, and A. F. Skarmeta, “Toward a Cybersecurity Certification Framework for the Internet of Things,” IEEE Security Privacy, vol. 17, no. 3, pp. 66–76, 5 2019.

[62]. ECSO, “European Cyber Security Certification a Meta-Scheme Approach v1.0,” 2017.

[63]. ETSI, Methods for Testing & Specification; Risk-based Security Assessment and Testing Methodologies, 2015.

[64]. HEAVENS, HAVENS: HEAling Vulnerabilities to ENhance Software Security and Safety – Project Proposal, 2012.

[65]. S. N. Matheu, J. L. Hernandez-Ramos, S. Perez, and A. F. Skarmeta, “Extending MUD Profiles through an Automated IoT Security Testing Methodology,” IEEE Access, pp. 1–20, 2019.

[66]. D. Barrera, I. Molloy, and H. Huang, “IDIoT - Securing the Internet of Things like it’s 1994.” [Online]. Available: http://arxiv.org/abs/1712.03623

[67]. ——, “Standardizing IoT Network Security Policy Enforcement,” in Workshop on Decentralized IoT Security and Standards, 2018.

[68]. Virtual Open Systems, “ISO 26262:2011 Certification.” [Online]. Available: http://www.virtualopensystems.com/en/company/ vosysmonitor-iso26262-asilc/

[69]. ARMOUR, “Deliverable D1.3: Experiments’ execution reporting and benchmarks,” 2018. [Online]. Available: https://ec.europa.eu/research/participants/documents/downloadPublic?documentIds=080166e5bab12def&appId=PPGMS

[70]. S. N. Matheu, J. L. Hernández-Ramos, A. F. Skarmeta, and G. Baldini, “A Survey of Cybersecurity Certification for the Internet of Things,” ACM Computing Surveys, vol. 53, no. 6, pp. 1–36, 2 2021. [Online]. Available: https://dl.acm.org/doi/10.1145/3410160

[71]. UNECE UN Regulation No. 155 - Cyber security and cyber security management system. https://unece.org/transport/documents/2021/03/standards/un-regulation-no-155-cyber-security-and-cyber-security

[72]. Booth, H., Rike, D., & Witte, G. A. (2013). The national vulnerability database (NVD): Overview.

[73]. R. Bachmann, A. D. Brucker, R. Bachmann, and A. D. Brucker, “Developing secure software,” Datenschutz und Datensicherheit - DuD, vol. 38, no. 4, pp. 257–261, 2014. [Online]. Available: http: //link.springer.com/10.1007/s11623-014-0102-0

[74]. W. Li, F. Le Gall, and N. Spaseski, “A Survey on Model-Based Testing Tools for Test Case Generation,” in Tools and Methods of Program Analysis, V. Itsykson, A. Scedrov, and V. Zakharov, Eds., vol. 779. Cham: Springer International Publishing, 2018, pp. 77–89.

[75]. M. Felderer, M. Büchler, M. Johns, A. D. Brucker, R. Breu, and A. Pretschner, “Chapter One - Security Testing: A Survey,” in Advances in Computers. Elsevier, 2015, vol. 101, pp. 1–51.

[76]. F. Bouquet, C. Grandpierre, B. Legeard, F. Peureux, N. Vacelet, and M. Utting, “A Subset of Precise UML for Model-Based Testing,” in Proceedings of the 3rd International Workshop on Advances in Model-Based Testing - A-MOST ’07. London, United Kingdom: ACM Press, 2007, pp. 95–104.

[77]. M. Felderer, B. Agreiter, P. Zech, and R. Breu, “A Classification for Model-Based Security Testing,” in VALID 2011, The Third International Conference on Advances in System Testing and Validation Lifecycle, 2011, pp. 109–114.

[78]. D. Xu, M. Tu, M. Sanford, L. Thomas, D. Woodraska, and W. Xu, “Automated Security Test Generation with Formal Threat Models,” IEEE Transactions on Dependable and Secure Computing, vol. 9, no. 4, pp. 526–540, 2012.

[79]. A. Cretin, B. Legeard, F. Peureux, and A. Vernotte, “Increasing the Resilience of ATC systems against False Data Injection Attacks using DSL based Testing,” in Doctoral Symposium ICRAT, 2018.

[80]. B. Legeard and A. Bouzy, “Smartesting CertifyIt: Model-Based Testing for Enterprise IT,” in 2013 IEEE Sixth International Conference on Software Testing, Verification and Validation. Luxembourg, Luxembourg: IEEE, 2013, pp. 391–397.

[81]. A. Vernotte, “Research Questions for Model-Based Vulnerability Testing of Web Applications,” in IEEE International Conference on Software Testing, Verification, and Validation Workshops, 2013.

[82]. J. Bozic and F. Wotawa, “Security Testing Based on Attack Patterns,” in IEEE International Conference on Software Testing, Verification, and Validation Workshops, 2014.

[83]. S. Yoo and M. Harman, “Regression Testing Minimization, Selection and Prioritization: A Survey,” Software Testing, Verification and Reliability, vol. 22, no. 2, pp. 67–120, 2012.

[84]. E. Fourneret, F. Bouquet, F. Dadeau, and S. Debricon, “Selective Test Generation Method for Evolving Critical Systems,” in Fourth International Conference on Software Testing, Verification and Validation Workshops, 2011.

[85]. M. Felderer and E. Fourneret, “A Systematic Classification of Security Regression Testing Approaches,” International Journal on Software Tools for Technology Transfer, vol. 17, no. 3, pp. 305–319, 2015.

[86]. L. Cseppento and Z. Micskei, “Evaluating Code-Based Test Input Generator Tools,” Software Testing, Verification and Reliability, vol. 27, no. 6, p.e1627, 2017.

[87]. B. Chess and J. West, Secure Programming with Static Analysis, 2013, vol. 53, no. 9.
[88]. N. Ayewah, D. Hovemeyer, J. D. Morgenthaler, J. Penix, and W. Pugh, "Using Static Analysis to Find Bugs," IEEE Software, vol. 25, no. 5, pp. 22–29, 2008. [Online]. Available: http://ieeexplore.ieee.org/document/4602670/
[89]. M. Bishop, "About Penetration Testing," IEEE Security & Privacy Magazine, vol. 5, no. 6, pp. 84–87, 2007.
[90]. J. Bau, E. Bursztein, D. Gupta, and J. Mitchell, "State of the Art: Automated Black-Box Web Application Vulnerability Testing," in 2010 IEEE Symposium on Security and Privacy. Oakland, CA, USA: IEEE, 2010, pp. 332–345.
[91]. ISECOM, "The Open Source Security Testing Methodology Manual (OSSTMMv3)," 2010.
[92]. M. Sutton, A. Greene, and P. Aminir, "Fuzzing: Brute Force Vulnerability Discovery." Pearson Education, 2007, pp. 1–51.
[93]. C. Chen, B. Cui, J. Ma, R. Wu, J. Guo, and W. Liu, "A Systematic Review of Fuzzing Techniques," Computers & Security, vol. 75, pp. 118–137, 2018.
[94]. M. Schneider, J. Grossmann, I. Schieferdecker, and A. Pietschker, "Online Model-Based Behavioral Fuzzing," in IEEE Sixth International Conference on Software Testing, Verification and Validation Workshops, 2013.
[95]. P. Tsankov, M. T. Dashti, and D. Basin, "SECFUZZ: Fuzz-testing Security Protocols," in Proc. of the 7th International Workshop on Automation of Software Test, 2012.
[96]. C. Miller and Z. Peterson, "Analysis of Mutation and Generation-Based Fuzzing," 2007.
[97]. W. Krenn, R. Schlick, S. Tiran, B. Aichernig, E. Jöbstl, and H. Brandl, "MoMut::UML model-based mutation testing for UML," in 2015 IEEE 8th International Conference on Software Testing, Verification and Validation, ICST 2015 - Proceedings, 2015.
[98]. F. Duchene, "Detection of Web Vulnerabilities via Model Inference assisted Evolutionary Fuzzing," Ph.D. dissertation, Grenoble University, [Online]. Available: https://hal.archives-ouvertes.fr/tel-01102325/
[99]. J. Bozic and F. Wotawa, "Model-based Testing - From Safety to Security,"STV Bozic, Wotawa, 2012.
[100]. S. Bekrar, C. Bekrar, R. Groz, and L. Mounier, "Finding Software Vulnerabilities by Smart Fuzzing," in Fourth IEEE International Conference on Software Testing, Verification and Validation, 2011.
[101]. Society of Automotive Engineers (SAE), "J3061 - Cybersecurity Guidebook for Cyber-Physical Vehicle Systems," p. 128, 2016. [Online]. Available: http://standards.sae.org/wip/j3061/
[102]. U.S. Department of Transportation, "Cybersecurity and Intelligent Transportation Systems: Best Practice Guide," 2019. [Online]. Available: https://rosap.ntl.bts.gov/view/dot/42461
[103]. ETSI, "ETSI TS 103 097. Intelligent Transport Systems (ITS); Security; Security header and certificate formats," 2017. [Online]. Available: https://forge.etsi.org/rep/ITS/ITS
[104]. CCRA, "Common Criteria, Assurance Continuity, CCRA Requirements. Version 2.1," 2012.
[105]. C. Zhou and S. Ramacciotti, "Common Criteria: Its Limitations and Advice on Improvement," ISSA Journal, 2011. [Online]. Available: https://www.difesa.it/SMD_/Staff/Reparti/II/CeVa/Pubblicazioni/Estere/Documents/CommonCriteria_ISSAJournal_0411.pdf
[106]. CAR 2 CAR Communication Consortium, "Protection Profile V2X Hardware Security Module," 2018.
[107]. European Union Agency for Network and Information Security (ENISA), "Cybersecurity Certification: Candidate EUCC Scheme," 2020. [Online]. Available: https://www.enisa.europa.eu/publications/cybersecurity-certification-eucc-candidate-scheme
[108]. I. Pekaric, C. Sauerwein, and M. Felderer, "Applying Security Testing Techniques to Automotive Engineering," in Proceedings of the 14th International Conference on Availability, Reliability and

Security. New York, NY, USA: ACM, 8 2019, pp. 1–10. [Online]. Available: https://dl.acm.org/doi/10.1145/3339252.3340329

[109]. P. Wooderson and D. Ward, “Cybersecurity Testing and Validation,” in SAE Technical Papers, vol. 2017-March, no. March 3 2017. [Online]. Available: https://www.sae.org/content/2017-01-1655/

[110]. A. Chattopadhyay and K. Y. Lam, “Autonomous vehicle: Security by design,” arXiv, 2018.

[111]. S. Marksteiner, N. Marko, A. Smulders, S. Karagiannis, F. Stahl, H. Hamazaryan, R. Schlick, S. Kraxberger, and A. Vasenev, “A Process to Facilitate Automated Automotive Cybersecurity Testing,” 2021. [Online]. Available: https://arxiv.org/abs/2101.10048

[112]. Y. Zhang, P. Shi, C. Dong, Y. Liu, X. Shao, and C. Ma, “Test and Evaluation System for Automotive Cybersecurity,” in 2018 IEEE International Conference on Computational Science and Engineering (CSE). IEEE, 10 2018, pp. 201–207. [Online]. Available: https://ieeexplore.ieee.org/document/8588239/

[113]. P. S. Oruganti, M. Appel, and Q. Ahmed, “Hardware-in-loop based Automotive Embedded Systems Cybersecurity Evaluation Testbed,” in Proceedings of the ACM Workshop on Automotive Cybersecurity. New York, NY, USA: ACM, 3 2019, pp. 41–44. [Online]. Available: https://dl.acm.org/doi/10.1145/3309171.3309173

[114]. D. S. Fowler, M. Cheah, S. A. Shaikh, and J. Bryans, “Towards a Testbed for Automotive Cybersecurity,” in Proceedings - 10th IEEE International Conference on Software Testing, Verification and Validation, ICST 2017, 2017, pp. 540–541.

[115]. D. S. Fowler, J. Bryans, S. A. Shaikh, and P. Wooderson, “Fuzz Testing for Automotive Cyber-Security,” in Proceedings - 48th Annual IEEE/IFIP International Conference on Dependable Systems and Networks Workshops, DSN-W 2018, 2018, pp. 239–246.

[116]. S. Mahmood, A. Fouillade, H. N. Nguyen, and S. A. Shaikh, “A Model-Based Security Testing Approach for Automotive Over-The-Air Updates,” in 2020 IEEE International Conference on Software Testing, Verification and Validation Workshops (ICSTW). IEEE, 10 2020, pp. 6–13. [Online]. Available: https://ieeexplore.ieee.org/document/9155945/

[117]. G. Bernabeu, E. Jaffuel, B. Legeard, and F. Peureux, “MBT for global platform compliance testing: Experience report and lessons learned,” in Proceedings - IEEE 25th International Symposium on Software Reliability Engineering Workshops, ISSREW 2014, 2014, pp. 66–70.

[118]. ECSO, “European Cyber Security Certification: A Meta-Scheme Approach v1.0,” 2017.

[119]. Khan, M. A., Jadoon, A., Haq, K. M. S., Mumtaz, S., & Rodrigues, J. (2019, May), “An overview of resilient and automatic model-based testing approaches for automotive industry,” In 2019 IEEE International Conference on Communications Workshops (ICC Workshops) (pp. 1-6).

[120]. Sommer, F., Kriesten, R., & Kargl, F. (2021, December), “Model-Based Security Testing of Vehicle Networks,” In 2021 International Conference on Computational Science and Computational Intelligence (CSCI) (pp. 685-691).

[121]. J. Lorrain, E. Fourneret, F. Dadeau, and B. Legeard, “MBeeTle - un outil pour la génération de tests à-la-volée à l’aide de modèles,” in Groupement De Recherche CNRS du Génie de la Programmation et du Logiciel, 2016. [Online]. Available: https://hal.archives-ouvertes.fr/hal-02472608

[122]. 3gpp, “3rd generation partnership project.” [Online]. Available: http: //www.3gpp.org/

[123]. S. Halder, A. Ghosal, and M. Conti, “Secure over-the-air software updates in connected vehicles: A survey,” Computer Networks, vol. 178, p. 107343, 9 2020. [Online]. Available: https://linkinghub.elsevier.com/retrieve/pii/S1389128619314963.

[124]. ISO/SAE 21434:2021, “Road vehicles — Cybersecurity engineering,” Available: https://www.iso.org/standard/70918.html, last accessed August 2, 2022.

[125]. DIAS Deliverable 4.1, “Security analysis, requirements identification and applicability of security solutions for tamper protection,” May 18, 2020. [Online]. Available: https://dias-project.com/sites/default /files/Deliverables/DIAS-D4.1-v1.0_Summary.pdf

[126]. DIAS Deliverable 3.1, “The market of cheating devices and testing matrix with a prioritization for testing of vehicle tampering technique combinations,” March 31, 2020. [Online]. Available: https://dias-project.com/sites/default/files/Deliverables/D3.1Cheating%20devices%20and %20testing%20matrix_0.pdf

[127]. DIAS Deliverable 4.2, “In-vehicular antitampering security techniques and integration,” March 31, 2021. [Online]. Available: https://dias-project.com/sites/default/files/D4.2%20In-vehicular %20antitampering%20security%20techniques%20and%20integration.pdf

[128]. DIAS Deliverable 4.3, “Distributed ledger technology (DLT) and cloud-based methods for the provisioning of certified data,” December 31, 2021. [Online]. Available: https://dias-project.com/sites/default/files/Deliverables/DIAS-D4.3_v1.0.pdf

[129]. Caring Caribou – A friendly car security exploitation tool for the CAN bus, https://github.com/CaringCaribou/caringcaribou, last accessed August 2, 2022.

[130]. INCA software products, https://www.etas.com/en/products/inca_software_products.php, Last accessed August 2, 2022.

[131]. DIAS Deliverable 3.2, “Status quo of critical tampering techniques and proposal of required new OBD monitoring functions,” December 23, 2020. [Online]. Available: https://dias-project.com/sites/default/files/Deliverables/D3.2%20-%20Status%20quo%20of%20critical %20tampering%20techniques%20and%20proposal%20of%20required%20new%20OBD%20monitoring%20functions.pdf

[132]. National Institute of Standards and Technology, “NIST.FIPS.186-4, Digital signature standard,” July 2013. [Online]. Available: https://nvlpubs.nist.gov/nistpubs/fips/nist.fips.186-4.pdf

[133]. National Institute of Standards and Technology, “NIST SP 800-108r1, Recommendation for Key Derivation Using Pseudorandom Functions,” [Online]. Available: https://nvlpubs.nist.gov /nistpubs/SpecialPublications/NIST.SP.800-108r1.pdf

[134]. National Institute of Standards and Technology, “Recommendation for Pair-Wise Key-Establishment Schemes Using Integer Factorization Cryptography,” [Online]. Available: https://nvlpubs.nist.gov/nistpubs/specialpublications/nist.sp.800-56br1.pdf

[135]. ISO/IEC 27001:2013, “Information technology — Security techniques — Information security management systems — Requirements,” https://www.iso.org/standard/54534.html. Last accessed August 2022.

[136]. Dobaj, J., Ekert, D., Stolfa, J., Stolfa, S., Macher, G., & Messnarz, R. (2021), “Cybersecurity Threat Analysis, Risk Assessment and Design Patterns for Automotive Networked Embedded Systems: A Case Study,” J. Univers. Comput. Sci., 27(8), 830-849.

[137]. Zhang, H., Pan, Y., Lu, Z., Wang, J., & Liu, Z. (2021), “A Cyber Security Evaluation Framework for In-Vehicle Electrical Control Units,” IEEE Access, 9, 149690-149706.

[138]. Schmittner, C., Ma, Z., Reyes, C., Dillinger, O., & Puschner, P. (2016, September), “Using SAE J3061 for automotive security requirement engineering,” In International Conference on Computer Safety, Reliability, and Security (pp. 157-170). Springer, Cham.

[139]. Macher, G., Armengaud, E., Brenner, E., & Kreiner, C. (2016, September), “A review of threat analysis and risk assessment methods in the automotive context,” In International Conference on Computer Safety, Reliability, and Security (pp. 130-141). Springer, Cham.

[140]. Radanliev, P., De Roure, D. C., Nicolescu, R., Huth, M., Montalvo, R. M., Cannady, S., & Burnap, P. (2018), “Future developments in cyber risk assessment for the internet of things,” Computers in industry, 102, 14-22.

[141]. Karahasanovic, A., Kleberger, P., & Almgren, M. (2017, November), “Adapting threat modeling methods for the automotive industry,” In Proceedings of the 15th ESCAR Conference (pp. 1-10).
[142]. Cui, J., & Sabaliauskaite, G. (2017), “On the alignment of safety and security for autonomous vehicles,” Proceedings of IARIA CYBER, 59-64.
[143]. SOIC test clips, [Online]. Available: https://www.farnell.com/datasheets/193574.pdf