Project acronym:	EVITA
Project title:	E-safety vehicle intrusion protected applications
Project reference:	224275
Programme:	Seventh Research Framework Programme (2007–2013) of the European Community
Objective:	ICT-2007.6.2: ICT for cooperative systems
Contract type:	Collaborative project
Start date of project:	1 July 2008
Duration:	36 months

Deliverable D2.3:
Security requirements for automotive on-board networks based on dark-side scenarios

Authors:	Alastair Ruddle, David Ward (MIRA); Benjamin Weyl (BMW Group Research and Technology GmbH); Sabir Idrees, Yves Roudier (EURECOM); Michael Friedewald, Timo Leimbach (Fraunhofer Institute ISI); Andreas Fuchs, Sigrid Gürgens, Olaf Henniger, Roland Rieke, Matthias Ritscher (Fraunhofer Institute SIT); Henrik Broberg (Fujitsu Services AB); Ludovic Apvrille, Renaud Pacalet, Gabriel Pedroza (Institut Télécom)
Reviewers:	Enno Kelling (Continental Teves AG & Co. oHG); Antonio Kung (Trialog); Marko Wolf (escrypt GmbH)

Dissemination level:	Public
Deliverable type:	Report
Version:	1.1
Submission date:	30 December 2009

Abstract

The objective of the EVITA project is to design, verify, and prototype an architecture for automotive on-board networks where security-relevant components are protected against tampering and sensitive data are protected against compromise. Thus, EVITA will provide a basis for the secure deployment of electronic safety aids based on vehicle-to-vehicle and vehicle-to-infrastructure communication. A key activity for the EVITA project is the capture of security requirements for the secure system architecture and associated software and hardware components based on a set of use cases and an investigation of security threat scenarios (dark-side scenarios). This document outlines the processes used to identify and evaluate security requirements, and details the results of their application to automotive on-board networks. It provides input to the secure on-board architecture design.

Terms of use

This document was developed within the EVITA project (see http://evita-project.org), co-funded by the European Commission within the Seventh Framework Programme (FP7), by a consortium consisting of a car manufacturer, automotive suppliers, security experts, hardware and software experts as well as legal experts. The EVITA partners are

- BMW Group Research and Technology GmbH,
- Continental Teves AG & Co. oHG,
- escrypt GmbH,
- EURECOM,
- Fraunhofer Institute for Secure Information Technology,
- Fraunhofer Institute for Systems and Innovation Research,
- Fujitsu Services AB,
- Infineon Technologies AG,
- Institut Télécom,
- Katholieke Universiteit Leuven,
- MIRA Ltd.,
- Robert Bosch GmbH and
- TRIALOG.

evita-feedback@listen.sit.fraunhofer.de

The comment form available from http://evita-project.org/deliverables.html may be used for submitting comments.

Contents

1 Introduction 1
1.1 Background 1
1.2 Purpose and scope 2
1.3 Organisation of the document 3

2 Security Engineering Process 4
2.1 Security properties 4
2.1.1 Informal description 4
2.1.2 Security modelling framework 6
2.2 Approach 14
2.3 System under investigation and its environment 15
2.4 Summary of use cases 16
2.5 System assets 18
2.6 Threat identification (dark-side scenarios) 18
2.7 Overview of risk analysis 19
2.8 Identification of security requirements 20
2.8.1 Overview 20
2.8.2 Abstract functional path approach 21
2.8.3 Detailed functional path and mapping approach 21

3 EVITA Security Requirements 24
3.1 Security objectives 24
3.2 Security requirements 24
3.2.1 Overview 24
3.2.2 Authenticity 25
3.2.3 Integrity 33
3.2.4 Controlled access 35
3.2.5 Freshness 36
3.2.6 Non-repudiation 37
3.2.7 Anonymity 38
3.2.8 Privacy 39
3.2.9 Confidentiality 42
3.2.10 Availability 43
3.3 Priority of security requirements 45

4 Conclusions 57

Appendix A – Glossary 58

Appendix B – Dark-side scenarios 63
B.1 Introduction 63
B.2 Attack motivations 64
B.2.1 Overview 64
B.2.2 Do psychological or physical harm to the driver 64
B.2.3 Gain information about the driver 65
B.2.4 Gain reputation as a hacker 65

B.2.5 Financial gain 65
B.2.6 Gain personal advantages (non financial) 66
B.2.7 Gain information about vehicle manufacturer 66
B.2.8 Harm the economy 67
B.2.9 Mass terrorism 67
B.3 Possible attacks – Combining attack motivations and use cases 68
B.3.1 Force Green Wave/Getting traffic lights green ahead of the attacker 68
B.3.2 Manipulate Speed Limits 70
B.3.3 Manipulate Traffic Flow 70
B.3.4 Simulate Traffic Jam 72
B.3.5 Tamper with Warning Message 72
B.3.6 E-Call 75
B.3.7 Engine DoS-Attack (Engine Refuse to Start) 75
B.3.8 Unauthorized Brake 76
B.3.9 Attacking Active Brake Function 79
B.3.10 Attacking E-Toll 80
B.4 Attack Trees Detailing Asset Attacks 82
B.4.1 Flashing per OBD 82
B.4.2 Head Unit Attack 84

Appendix C – Threat and risk analysis 85
C.1 Analysis methodology 85
C.1.1 Introduction 85
C.1.2 Notion of severity 85
C.1.3 Notion of probability of occurrence of successful attack (attack potential) 86
C.1.4 Estimating risk 89
C.1.5 Requirements for countermeasures 91
C.2 EVITA Risk Analysis 92
C.2.1 Attack potential 92
C.2.2 Attack active brake function 97
C.2.3 Tamper with warning message 98
C.2.4 Attacking E-Call 99
C.2.5 Unauthorized brake 100
C.2.6 Attack E-Toll 101
C.2.7 Green light ahead of attacker 102
C.2.8 Manipulate speed limits 103
C.2.9 Simulate traffic jam 105
C.2.10 Manipulate traffic flow 105
C.2.11 Engine denial of service 108
C.3 Summary and conclusions 109

Appendix D – Identifying security requirements 111
D.1 Abstract functional path approach 111
D.1.1 Abstract functional system model 111
D.1.2 Security Requirements Engineering Process 113

D.2 Detailed functional path and mapping approach 114
D.2.1 Methodology 114
D.2.2 Classification of attacks 117
D.2.3 SysML based security requirements 119
D.2.4 Functional and mapping views of use cases 130

References 138

List of figures

Figure 1 Generalised architecture of automotive on-board networks 15
Figure 2 EVITA use case reference architecture 16
Figure 3 Generic attack tree structure 63
Figure 4 Attack tree 1: Force green lights ahead of attacker 69
Figure 5 Attack tree 2: Manipulate speed limits 70
Figure 6 Attack tree 3: Manipulate traffic flow 71
Figure 7 Attack tree 4: Simulate traffic jam 73
Figure 8 Attack tree 5: Tamper with warning messages 74
Figure 9 Attack tree 6: Attacking E-Call 76
Figure 10 Attack tree 7: Engine refuses to start 77
Figure 11 Attack tree 8: Unauthorized brake 78
Figure 12 Attack tree 9: Attack active brake function 79
Figure 13 Attack tree 10: Prevent driver from passing toll gate 80
Figure 14 Attack tree 11: Increase driver's toll bill 81
Figure 15 Attack tree 12: Reduce driver's toll bill 81
Figure 16 Attack tree 13: Compromise driver privacy 82
Figure 17 Attack tree 14: OBD flashing attack 83
Figure 18 Attack tree 15: Head unit attack 84
Figure 19 Abstract functional system model pattern 112
Figure 20 Abstract functional system model instance 113
Figure 21 Identification of security requirements 115
Figure 22 SysML Diagram Description 122
Figure 23 SysML Environment Related Security Requirements 125
Figure 24 SysML Availability Security Requirements 127
Figure 25 SysML Privacy Security Requirements 128
Figure 26 SysML Fake Command Security Requirements 129
Figure 27 SysML Flashing Security Requirements 131
Figure 28 SysML Braking DoS Security Requirements 132
Figure 29 Functional view of Safety Reaction Active Brake 134
Figure 30 Mapping view of Safety Reaction Active Brake 135
Figure 31 Functional view of Flashing per OBD 136
Figure 32 Mapping view of Flashing per OBD 137

List of tables

Table 1 Generic security threats and security objectives 19
Table 2 Combined risk graph for safety-related (C≥1) and non-safety (C=1) security threats 46
Table 3 Summary findings of risk analysis 47
Table 4 Proposed severity classification scheme for security threats 86
Table 5 Rating of aspects of attack potential 88
Table 6 Rating of attack potential and attack probability 89
Table 7 Tabular representation of key elements of an attack tree 89
Table 8 Attack tree of Table 7 augmented with risk analysis parameters 90
Table 9 Proposed security risk graph for non-safety security threats (privacy, financial and operational) 90
Table 10 Classification for controllability of safety hazards 91
Table 11 Proposed security risk graph for safety-related security threats 91
Table 12 Evaluation of required attack potential for asset attacks identified from attack trees 93
Table 13 Risk analysis for “Attack Active Brake Function” 98
Table 14 Risk analysis for “Tamper with Warning Message” 99
Table 15 Risk analysis for “Attacking E-Call” 100
Table 16 Risk analysis for “Unauthorized brake” 101
Table 17 Risk analysis for “Attacking E-Toll” 102
Table 18 Risk analysis for “Green light ahead of attacker” 103
Table 19 Risk analysis for “Manipulate speed limits” 104
Table 20 Risk analysis for “Simulate traffic jam for target car” 106
Table 21 Risk analysis for “Manipulate traffic flow” 107
Table 22 Risk analysis for “Engine denial of service” 109
Table 23 Views on activities 116
Table 24 Description of SysML symbols 123
Table 25 Security Requirements Coverage – Attack Trees 8 and 9 133

List of abbreviations

ABS	Anti-lock Braking System
CSC	Chassis Safety Controller
CPU	Central Processing Unit
CU	Communication Unit
DoS	Denial of Service
DSRC	Digital Short Range Communication
ECU	Electronic Control Unit
EEPROM	Electrically Erasable Programmable Read-Only Memory
ESP	Electronic Stability Program
GPS	Global Positioning System
GSM	Global System for Mobile Communications
HMI	Human Machine Interface
HU	Head Unit
IPR	Intellectual Property Rights
ITS	Intelligent Transport System
OBD	On-Board Diagnostics
PoI	Point of Interest
PSAP	Public Safety Access Point
PTC	Powertrain Controller
RAM	Random Access Memory
RSU	Road Side Unit
TOE	Target of Evaluation
UMTS	Universal Mobile Telecommunications System
USB	Universal Serial Bus

Document history

Version	Date	Description
1.0	31/03/2009	First issue of deliverable
1.1	30/12/2009	Disposition of the comments received at the first EVITA project review

1 Introduction

1.1 Background

Future visions of road transportation include the networked vehicles and intelligent transport systems (ITS) that will enhance the safety of drivers and other road users, minimize pollution and maximize the efficiency of travel. The nature and interests of the stakeholders involved in future road transport systems therefore include:

- **vehicle users** – safe and efficient driving, valid financial transactions, personal privacy, protection of personal data;
- **other road users** – safe and efficient transport;
- **vehicle/sub-system manufacturers** – successful and affordable satisfaction of customer expectations, protection of IPR;
- **ITS system operators** – safe and efficient operation of systems, valid financial transactions, protection of user data;
- **civil authorities** – safe and efficient transportation networks, reliable financial transactions, data protection.

For the networked vehicles and intelligent transport systems (ITS) envisaged for the future, unauthorized access to vehicle or personal data may become possible, while the corruption of data or software could result in anomalies in vehicle function or traffic behaviour. Potential threat agents and their objectives may include:

- **dishonest drivers** – avoid financial obligations, gain traffic advantages;
- **hackers** – gain/enhance reputation as a hacker;
- **criminals and terrorists** – financial gain, harm or injury to individuals or groups;
- **dishonest organisations** – driver profiling, industrial espionage, sabotage of competitor products;
- **"rogue states"** – achieve economic harm to other societies.

Security functional requirements for information systems are broadly categorized into three types [1]:

- **confidentiality** – prevention of unwanted/unauthorized disclosure of data;
- **integrity** – prevention of unwanted/unauthorized alteration or creation of data;
- **availability** – prevention of unwanted/unauthorized loss of data or access to data.

The EVITA project is concerned specifically with on-board networks within individual vehicles, rather than the wider ITS systems. In future road transport scenarios, breaches in the security of vehicle information or functions could lead to possible issues for stakeholders in four main areas:

- **privacy** – unwanted/unauthorized acquisition of data relating to vehicle/driver activity, vehicle/driver identity data, or vehicle/sub-system design and implementation;

- **financial** – unwanted/unauthorized commercial transactions, or access to vehicle;
- **operational** – unwanted/unauthorized interference with on-board vehicle systems or Car2X communications that may impact on the operational performance of vehicles and/or ITS systems (without affecting physical safety);
- **safety** – unwanted/unauthorized interference with on-board vehicle systems or Car2X communications that may impact on the safe operation of vehicles and/or ITS systems.

An important implication of this is that a subset of security issues may also impact on functional safety. Engineering processes that aim to ensure functional safety properties in programmable automotive systems are described in ISO/DIS 26262 [2] and the MISRA safety analysis guidelines [3]. These methods are based on the process industry standard IEC 61508 [4], with adaptations to reflect the particular issues associated with automotive applications. Thus, there is a need to ensure that security issues with safety implications also meet the requirements of safety engineering processes. In addition, there is also a need to adapt the approaches defined in the IT security evaluation standard ISO/IEC 15408 [1] in order to address the particular issues of automotive applications, such as the possibility that a security threat may also have safety implications.

In order to define security and safety requirements for a system it is necessary to have an understanding of the operating environment and intended behaviour of the system. This is achieved through the specification of use cases for automotive on-board networks [5]. These use cases may themselves suggest a number of security-related user requirements. However, the use cases also provide the basis for investigating a number of "dark scenarios" (threats), which are intended to establish ways in which the system could become a target for malicious attacks. The security issues identified from the dark scenarios are likely to include examples that also have safety implications.

1.2 Purpose and scope

The aim of the security requirements analysis is to derive, justify and prioritise IT security requirements and IT security related safety requirements for automotive on-board networks. Only run-time requirements are considered. How to gain assurance during the development process that these requirements are met at run time is out of scope of this report.

The inputs to the security requirements analysis are example use cases [5], dark-side scenarios, and the state of the art in standards and research. These inputs are viewed as the rationale for the requirements. The use cases require certain security functions in order to protect identified assets within the use case scenarios. The use cases also provide constraints and assumptions, such as performance constraints for the security functions. Security risk analysis of the threats identified in the dark-side scenarios will be documented as the rationale for the security objectives and security requirements. Traceability between the threats, objectives and requirements is accomplished by a structured approach.

The security requirements then provide inputs to the secure on-board architecture design, to the model-based verification of on-board architecture and protocol specifications, to the security architecture implementation as well as to the analysis of legal aspects presented in forthcoming EVITA deliverables.

This report defines a process for identifying vehicle security requirements, for assessing the relative risks of possible threats, and for addressing the subset of these security require-

ments that may be safety related. This process is then piloted to formulate requirements for the countermeasures needed to reduce the vulnerability of the vehicle's on-board architecture to threats that may lead to possible safety concerns and risks to assets. The results of its application are documented in this report. In addition, this document also details the translation of these requirements into a semi-formal requirements specification, which will provide the starting point for the model-based verification. It should be noted, however, that the specification of security requirements is an iterative process, rather than a completely self-contained activity. It is anticipated that the security requirements will evolve in the course of the design process, as is the case with other types of requirements (operational, functional safety etc.). The security requirements and priorities may also shift slightly when other use cases of automotive on-board networks and new threats are taken into consideration.

1.3 Organisation of the document

The process developed for deriving the security requirements analysis is outlined in Section 2. That section provides a wider description of the security engineering process, including aspects such as use case definition and dark-side scenarios investigation. The security requirements derived from the use cases and risk analysis of the attack trees are detailed in Section 3. Section 4 details the conclusions drawn from this activity.

A glossary detailing key terminology used in this document can be found in Appendix A – Glossary. Appendix B – Dark-side scenarios – summarizes relevant results (including attack trees) from the dark-side scenarios investigation, while Appendix C – Threat and risk analysis – provides details of the risk analysis approach and the results of its application to the attack trees. Appendix D – Identifying security requirements – contains a more detailed description of the approaches used to derive security requirements from the use cases, attack trees and risk analysis.

2 Security Engineering Process

2.1 Security properties

2.1.1 Informal description

2.1.1.1 Introduction

Before detailing the security engineering process, we introduce classes of security requirements that are relevant for automotive on-board networks. The (informal) explanations reflect the way these concepts are generally understood.

2.1.1.2 Data origin authenticity

A data origin authenticity property applies to a quantum of information and a claimed author. The property is satisfied when the quantum of information truly originates from the author. The property can be made more specific by providing an observation of the quantum of information (defined, e.g., by a time and a location in the system). The author can also be constrained by adding a time and/or a place of creation of the quantum of information. Note that in most security-oriented frameworks data origin authenticity implies integrity.

2.1.1.3 Integrity

An integrity property applies to a quantum of information between two observations (defined, e.g., by a time and a location in the system). The property is satisfied when the quantum of information has not been modified between the two observations. It guarantees for instance that the content of a storage facility has not been modified between two given read operations, or that a message sent on a communication channel has not been altered during its journey.

2.1.1.4 Controlled access (authorization)

A controlled access property or requirement applies to a set of actions and/or information and a set of authorized entities. The property is guaranteed if the specified entities are the only entities that can perform the actions or access the information. The property can be further detailed with time constraints on the period of authorization.

Controlled access is needed to ensure that stakeholders only have access to information and functions that they are authorized to access as appropriate to their expected activities.

2.1.1.5 Freshness

A freshness property or requirement applies to a quantum of information, a receiving entity and a given time. The property is satisfied if the quantum of information received by the entity at the given time is not a copy of the same information received by the same or another entity in the past. Ensuring freshness can be used to prevent replay attacks.

2.1.1.6 Non-repudiation

A non-repudiation property or requirement applies to an action and an entity performing the action. The non-repudiation of the action is guaranteed if it is impossible for the entity that performed the action to claim that it did not perform the action. This property can be further detailed with a set of entities for which the action needs to be undeniable, with a time limit, etc.

There may be specific legal requirements for non-repudiation. However, non-repudiation may also be introduced for convenience, for example, as an aid in providing evidence or proving liability.

2.1.1.7 Privacy/anonymity

A privacy property or requirement applies to an entity and a set of information. Privacy is guaranteed if the relation between the entity and the set of information is confidential. Anonymity, for instance, is the property that the relation between an entity and its identity is confidential.

Privacy is frequently a major concern when the entity involved is an individual or a vehicle owned by an individual. For example, an adversary constantly recording the location of a vehicle and knowing the identity of the driver may be considered as violating the driver's privacy with respect to her movements.

Privacy requirements are needed to ensure that the anonymity of stakeholders and confidentiality of their sensitive information are assured. Sensitive information introduced by the application shall be identified. For users, sensitive information may include (but is not limited to) the following:

- identity of a specific car and/or driver,
- current location of a specific car and/or driver,
- past locations of a specific car and/or driver,
- properties of the vehicle that can be used for tracking a specific car and/or driver (e.g. car manufacturer, model, colour),
- behaviour of a specific car and/or driver (e.g. number of critical situations, speeding),
- records of telephone calls, internet activity, email messages, account information and driving characteristics,
- identity of specific cars and/or drivers involved in particular C2X transactions.

For vehicle manufacturers and system suppliers, sensitive information may include (but is not limited to) the following:

- identity of a specific car,
- car manufacturer and model,
- design information (algorithms, control parameters),
- performance data.

Privacy requirements must be made consistent with potentially conflicting requirements for identification, auditing, non-repudiation and jurisdictional access, which may require users to be identified and information about their interactions to be stored.

2.1.1.8 Confidentiality

A confidentiality property applies to a quantum of information and a set of authorized entities. The property is satisfied when the authorized entities are the only ones that can know the quantum of information. Privacy relies on confidentiality and can be considered as a special case of confidentiality.

2.1.1.9 Availability

An availability property or requirement applies to a service or a physical device providing a service. The property is satisfied when service is operational. Denial of service attacks aim at compromising the availability of their target. The property can be further detailed with the specification of a period during which the availability is required and of a set of client entities requesting the availability.

2.1.2 Security modelling framework

2.1.2.1 Overview

In the following the Security Modelling Framework (SeMF) of Fraunhofer SIT is informally described. It allows describing more abstract security requirements than the concepts of Section 2.1 and is useful when modelling systems at high levels of abstraction. This framework will be used in Section 3 to specify high-level security requirements that are relevant to automotive on-board networks.

The underlying formal model describes system behaviours as (sets of) sequences (traces) of actions. These actions in turn are mostly associated with agents in the systems (system entities or stakeholders). The actions describe what can happen in the system and have to be carefully chosen in order to be able to express all desired system properties. System specification based on sequences of actions is very common, but for security properties additional information is required:

- First, satisfaction of security properties depends on the agents' view of the system. In SeMF, this view has to be specified for each system entity for which a security property shall hold.
- Second, for each agent the knowledge about the global system has to be part of the system specification. For example, trust in underlying security mechanisms, such as cryptographic algorithms, is described as knowledge about the system.

In the following sub-sections an informal description of SeMF is given with the objective to understand the properties that are being specified for the EVITA use cases. For the formal framework, we refer the reader to [6][7][8] and to forthcoming EVITA deliverables.

Throughout this section we will use a simple example to illustrate our explanations. Our simple example system has four different agents: users *U* and *V*, and service providers *S* and *T*. Service providers send offers to users by using actions *sOffer(sp,user,price)*; these are received by the users with action *rOffer(user,sp,price)*. Users can then order (action *sOrder(user,sp,price)*), and the service providers can receive the orders (action *rOrder(sp,user,price)*). For simplicity, price can have two different values: *cheap* and *exp*.

Note that this example is just to illustrate our approach and does not restrict the framework to communication scenarios. Any type of actions with appropriate type and number of parameters can be used. Possible examples relevant to EVITA include *Sense(sensor,EmergencyBrakeMessage)*, *Send(otherCar,message)*, *brake(Brake-Controller(Car))*, etc. If some agent performs an action we usually denote this agent using the first parameter of this action.

A system and its security properties are specified by those sequences of actions that can happen in the system. In our example system, a possible sequence of actions could be *sOffer(S,U,cheap) rOffer(U,S,cheap) sOrder(U,S,cheap) rOrder(S,U,cheap)*. Another possible sequence could be *sOffer(S,U,cheap) sOffer(S,V,exp) rOffer(U,S,cheap)*. However, in our system we would probably not allow a message to be received without having been sent, thus *sOffer(S,U,cheap) rOrder(T,V,exp)* would not be a possible sequence for the system.

2.1.2.2 Agents' view and knowledge of global system behaviour

2.1.2.2.1 General

Security properties can only be satisfied relative to particular sets of underlying system assumptions. Examples include assumptions regarding cryptographic algorithms, secure storage, and trust in the correct behaviour of agents or reliable data transfer. Relatively small changes in these assumptions can result in huge differences concerning satisfaction of security properties. Every model for secure systems must address these issues.

In order to provide the required flexibility, we extend the system specification by two components:

- the agents' initial knowledge about the global system behaviour and
- the *agents' views*.

The knowledge about the system consists of all traces that an agent initially considers possible, i.e. all traces that do not violate any system assumptions. The local view of an agent specifies which parts of the system behaviour the agent can actually see. In the following subsections, these two components and their relations are explained in more detail.

2.1.2.2.2 Agents' initial knowledge

For any agent *P*, W_P denotes its knowledge about the global system behaviour and contains those sequences of actions that *P* considers to be principally possible in the system. W_P is considered to be part of the system specification. We may assume for example that a message that was received must have been sent before. Thus an agent's W_P will contain only those sequences of actions in which a message is first sent and then received. As another example,

all sequences of actions included in W_P in which a digital signature is received and verified by using some agent Q's public key will contain an action where Q generated this signature.

Care must be taken when specifying the sets W_P for all agents P in order to avoid specifying properties that are desirable but not guaranteed by verified system assumptions. For example, in a scenario where we assume one-time passwords are used, if P trusts Q, W_P contains only those sequences of actions in which Q sends a certain password only once. However, if Q cannot be trusted, W_P will also contain sequences of actions in which Q sends a password more than once.

2.1.2.2.3 Agents' local view

The set W_P describes what P knows initially. However, in a running system P can learn from actions that have occurred. Satisfaction of security properties obviously also depends on what agents are able to learn. After a sequence of actions w of the system has happened, every agent can use its *local view* of w to determine the sequences of actions it considers to be possible. For any system specification, the local view of the agents has to be specified appropriately. One simple local view is that agents only see their own actions. In this case, user U's local view of the sequence of actions

$w =$ *sOffer(S,U,cheap) rOffer(U,S,cheap) sOrder(U,S,cheap) rOrder(S,U,cheap)*

is *rOffer(U,S,cheap) sOrder(U,S,cheap)*. In some systems, however, it may be possible for an agent to also notice actions such as send and receive performed by other agents, but not to be able to actually recognize the messages that are being sent and received. In this case, U's local view of w would be *sOffer(S,U) rOffer(U,S,cheap) sOrder(U,SP,cheap) rOrder(S,U)*.

Let us consider now a specific sequence of actions w. Since an agent P only sees parts of it, there are other sequences in the system that look the same for P, i.e. that result in the same local view for P. In the case where agents only see their own actions, for example, U's view of w is *rOffer(U,S,cheap) sOrder(U,S,cheap)*. But this is also U's local view of the sequence

$w_2 =$ *sOffer(T,V,exp) sOffer(S,U,cheap) rOffer(V,T,exp) rOffer(U,S,cheap) sOrder(U,S,cheap) rOrder(S,U,cheap)*,

and of many other possible sequences of actions in the system.

Depending on its knowledge about the system, underlying security mechanisms and system assumptions, an agent does not consider all sequences that look the same as w to be possible. Thus it can use its knowledge to reduce this set: after w has happened, agent P considers only those sequences of actions that look like w with respect to its local view and that are at the same time included in its initial knowledge W_P to be possible. Although the sequence

$w_3 =$ *rOffer(V,S,cheap) sOffer(S,U,cheap) rOffer(U,S,cheap) sOrder(U,S,cheap) rOrder(S,U,cheap)*

looks the same as

sOffer(S,U,cheap) rOffer(U,S,cheap) sOrder(U,S,cheap) rOrder(S,U,cheap)

for U, U does not consider w_3 possible after w has happened because it knows that a message that has been received must have been sent before, and w_3 violates this assumption.

The set of sequences of actions that a specific agent considers possible after a specific sequence of actions has happened is the basis for the security properties described in the next subsections.

2.1.2.3 Authenticity

2.1.2.3.1 Concept

In the context of sequences of actions, authenticity is a property of a particular action. This property only makes sense from the viewpoint of a particular agent: while one agent wants a specific message authentically to originate from a specific sender, for example, another agent might not even know that the message exists. Thus we call a particular action *a* authentic for agent *P* if in all sequences that *P* considers possible after a sequence of actions *w* has happened, *a* must have happened some time in the past. In other words, all sequences of actions that look like *w* for *P* with respect to its local view and that are also contained in *P*'s initial knowledge W_P, must contain an action *a*.

By extending this definition to a set of actions being authentic for *P* if one of the actions in the set is authentic for *P*, we gain the flexibility that *P* does not necessarily need to know all parameters of the action in order to be authentic. For example, a message may consist of one part protected by a digital signature and another irrelevant part without protection. Then, the recipient can know that the signer has sent a message containing the signature, but the rest of the message is not authentic. Therefore, in this case, the set of actions to be authentic for *P* comprises all messages containing the relevant signature and arbitrary other message parts.

A possible authenticity requirement for our example system could be that having received an order presumably made by user *U*, *S* wants the respective send action to have authentically been performed by *U*. In Section 3, we will use particular instantiations of authenticity:

authentic(action1, action2, agent)

denotes the property that each time *agent* has performed *action2*, *action1* is authentic for her. Concrete local views and initial knowledge of agents will be specified later in the project since they can only be specified when the mechanisms to provide the security properties are identified.

The concept of authenticity is a generalization of data origin authenticity as explained in Section 2.1: A data origin authenticity property applies to a quantum of information and a claimed author. The property is satisfied when the quantum of information truly originates from the author.

Once the actions are fixed in which the quantum of information is generated and generation by a particular author is claimed, respectively, data origin authenticity can be expressed by requiring that for each entity for which the property shall hold, each time they perform the action in which generation of the information by a particular author is claimed, the generation action must have been authentically performed by this author.

2.1.2.3.2 Proof of authenticity – non-repudiation

Some actions do not only require authenticity but also need to provide a proof of authenticity. If agent *P* owns a proof of authenticity for a set of actions, it can provide this proof to other

agents (e.g. by sending it) who can in turn take possession of the proof (e.g. by receiving it) and can then be convinced of the action's authenticity. So for the definition of proof of authenticity of a specific set of actions the following aspects are relevant:

1. Once an agent *P* has performed an action that brings it into possession of the proof, *P* itself must be convinced of the authenticity of the action the proof refers to.
2. It must be possible for *P* to forward the proof to any other agent of the system (e.g. by sending it).
3. Each other agent, when getting forwarded the proof by *P*, must be able to perform an action that results in owning it (e.g. receiving). The agent must then be convinced of the authenticity of the action the proof refers to.

In the formal definition we use three sets of action: the set of actions that shall be authentic, the set of actions to forward a proof, and the set of actions that result in owning the proof. The definition is the simplest one. Other definitions consider the fact that proofs of authenticity can get lost or deleted by the agents, or that although a proof has been forwarded it is never actually received.

The concept of proof of authenticity is in line with the concept of non-repudiation introduced in Section 2.1. More specific proofs of authenticity can be defined, for example, to capture the case in which it is necessary to reduce the set of agents for which proofs shall exist, or to allow for loss of proofs, etc. Particular non-repudiation requirements as explained in the following examples are taken again from our example system:

- For a service provider, non-repudiation of origin of the order is provided if the service provider, having performed the *rOrder* action, owns a proof of authenticity of the respective *sOrder* action having been performed by a specific user.
- A user might require non-repudiation of the receipt of its order by the service provider. This can be accomplished by introducing one more actions that model the sending of a receipt performed by the service provider. Then non-repudiation of receipt is the requirement that when the user has received this acknowledgement message by a specific service provider, she owns a proof that this message is authentically sent by this service provider.
- non-repudiation of submission and delivery require a third party, thus our example system would need to be extended to cover this. We will not consider these requirements any further as they are not relevant in EVITA.

In Section 3, we will use particular instantiations of proof of authenticity to express particular non-repudiation requirements: *non-rep-origin (action1,action2,agent)* denotes the property that each time *agent* has performed *action2*, *action1* is authentic for her and she owns a proof of authenticity for *action1*. Again, concrete local views and initial knowledge of agents will be specified later in the project since they can only be specified when the mechanisms to provide security properties are identified.

2.1.2.3.3 Authenticity with respect to a phase

In many cases it is not only necessary to know *who* has performed a particular action, but also the specific *time* of the action. As our specification does not model any real-time properties,

time is modelled in terms of relations between actions in a sequence. However, discrete time can be included explicitly by introducing a clock.

In order to capture time, we use the definition of a *phase* provided in [9]. A phase is a subsystem of a system that is closed with respect to continuation of actions. Generally a phase can be a very complex part of the system. Phases often have well defined start and end actions. However for the purposes of EVITA it is sufficient to consider only those phases that have one start action and usually also only one end action (unless we want to model timeouts, for example). Closure with respect to continuation means that when the start action of a phase is performed, all actions that can continue this sequence of actions must be contained in the phase (i.e. cannot happen outside) until (one of) the last action(s) is reached.

This concept together with the concept of authenticity matches perfectly, for example, the idea of authentication protocols. An agent sends out a challenge (a random number) that starts the phase. Everything that can happen after that is part of the phase until the agent finally receives the challenge along with e.g. some digital signature, which ends the phase. We say that a set of actions is authentic for an agent P after a sequence of actions w with respect to a phase V if the set of actions is authentic for P and has happened within phase V. That is, in all sequences of actions P considers possible after w has happened, some action of this set must have happened within the phase V.

Integrity as explained in Section 2.1.1.3 can be expressed in terms of authenticity within a phase. In Section 2.1.1.3, we have defined integrity as a property that holds when a quantum of information has not been modified between two particular observations which can be determined for instance by a particular time and location in the system. This means that we have two actions that instantiate these observations, for example the consecutive reading of some data by some entity. We then require that each time the data be read, it must not have changed with respect to the last read action. In terms of authenticity within a phase, we define a phase to start with a read action by a particular entity and then require that for this entity, when she performs the next read action reading *data*, the first read action must have processed *data* and must have authentically occurred within the phase.

2.1.2.4 Confidentiality

2.1.2.4.1 Concept

Confidentiality in SeMF essentially formalizes the concept that an agent P, having monitored a specific sequence of actions w, cannot tell from its local view of what it has monitored and from its initial knowledge W_P about the system which was the specific parameter used in a specific action (or actions) of w, even if the set of possible parameters is known. Various aspects are included in our definition.

1. First, we have to consider agent P's local view of the sequence w it has monitored and the set of sequences that are identical for P with respect to its local view.
2. Second, P can discard some of the sequences from this set, depending on its knowledge of the system and the system assumptions, all formalized in W_P. For example, dependencies may exist between parameters in different actions known by P, such as a credit card number remaining the same for a long time, in which case P considers only those sequences of actions possible in which an agent always uses the same credit card number.

3. We need to identify the dependencies between actions that agents are generally allowed to know. We may want to allow them to know that credit card numbers for Master and Visa, respectively, remain the same, but we may not allow them to know that agents use a specific credit card (either Master or Visa) for ordering specific services.
4. Finally, we need to identify what it is exactly that we want to be confidential. There are several possibilities. We may be interested in the confidentiality of one specific parameter in one specific action, regardless of whether or not *P* knows dependencies between actions concerning this parameter. We may on the other hand want to formalize that *P* is not allowed to know certain dependencies between some actions that use this parameter, regardless of whether or not *P* knows the actual parameter that is used in a specific action. We may further want to formalize a combination of these two requirements.

Essentially, in our definition, parameter confidentiality is captured by requiring that for the action(s) having happened in *w* that shall be confidential for agent *P* with respect to some parameter *p*, all possible (combinations of) values for *p* occur in the set of sequences of actions *P* considers possible after *w*.

2.1.2.4.2 Confidentiality example

We again use our simple example introduced at the beginning. We want the system to meet the following requirement: *V* is not allowed to know which price *S* offered to *U*. This requirement already addresses point 4 in the above list as it specifies that we are only interested in the confidentiality of a specific parameter in one single action, namely in *sOffer(S,U,p)*.

Concerning point 1, let us assume that *V* can only see its own actions. Further, agents initially know that a message received must have been sent. *V* additionally knows that *U* only orders *cheap* and that *U* only orders after having received an offer. This addresses point 2 in the above list. As to the question of which are the allowed dependencies (point 3 above), we allow agents to know that a receive action must be preceded by a send. Let us assume the following sequence of actions has happened:

w = *sOffer(S,U,cheap) rOffer(U,S,cheap) sOffer(S,V,exp) rOffer(V,S,exp) sOrder(U,S,cheap).*

V's local view of *w* is *rOffer(V,S,exp)* since *V* does not see the actions of the other agents. Sequences of actions that look identical for *V* with respect to its local view include all combinations of sending and receiving offers and orders performed by *U*, *S* and *T* with its own action *rOffer(V,S,exp)* somewhere in between, and could include the following examples:

sOffer(S,U,cheap) rOffer(U,S,cheap) sOffer(S,V,exp) rOffer(V,S,exp)
sOffer(S,U,cheap) rOffer(U,S,exp) sOffer(S,V,exp) rOffer(V,S,exp)
sOffer(S,U,exp) rOffer(U,S,cheap) sOffer(S,V,cheap) rOffer(V,S,exp)
sOffer(S,U,exp) rOffer(U,S,exp) sOffer(S,V,exp) rOffer(V,S,exp)
sOffer(S,U,cheap) rOffer(U,S,cheap) sOffer(S,V,exp) rOffer(V,S,exp) sOrder(U,S,cheap) sOffer(S,U,cheap)
sOffer(S,U,cheap) rOffer(U,S,cheap) sOffer(S,V,exp) rOffer(V,S,exp) sOrder(U,S,cheap) sOffer(S,U,exp)
sOffer(S,U,exp) rOffer(U,S,exp) sOffer(S,V,exp) rOffer(V,S,exp) sOrder(U,S,exp) sOffer(S,U,cheap)
sOffer(S,U,exp) rOffer(U,S,exp) sOffer(S,V,exp) rOffer(V,S,exp) sOrder(U,S,exp) sOffer(S,U,exp)
sOffer(S,U,exp) rOffer(U,S,exp) sOffer(S,V,exp) rOffer(V,S,exp) sOffer(S,U,cheap)
sOffer(S,U,exp) rOffer(U,S,exp) sOffer(S,V,exp) rOffer(V,S,exp) sOffer(S,U,exp)

Now *V* can use its initial knowledge to disregard those sequences that violate this knowledge. Hence sequences are disregarded that contain a receive action without the respective send

action before. Further, those sequences in which *U* orders *exp* are disregarded. The resulting set of sequences contains, for example:

sOffer(S,U,cheap) rOffer(U,S,cheap) sOffer(S,V,exp) rOffer(V,S,exp)
sOffer(S,U,exp) rOffer(U,S,exp) sOffer(S,V,exp) rOffer(V,S,exp)
sOffer(S,U,cheap) rOffer(U,S,cheap) sOffer(S,V,exp) rOffer(V,S,exp) sOrder(U,S,cheap) sOffer(S,U,cheap)
sOffer(S,U,cheap) rOffer(U,S,cheap) sOffer(S,V,exp) rOffer(V,S,exp) sOrder(U,S,cheap) sOffer(S,U,exp)
sOffer(S,U,exp) rOffer(U,S,exp) sOffer(S,V,exp) rOffer(V,S,exp) sOffer(S,U,cheap)
sOffer(S,U,exp) rOffer(U,S,exp) sOffer(S,V,exp) rOffer(V,S,exp) sOffer(S,U,exp)

We now formalize that we are only interested in the parameter actually offered by *S* to *U*, disregarding the dependencies agents might know. This results in a set of sequences of actions containing only *sOffer(S,U,cheap)* and *sOffer(S,U,exp)* in arbitrary combinations. This reduces the above set (which is of course only a subset of all sequences deduced so far) to:

sOffer(S,U,cheap)
sOffer(S,U,exp)
sOffer(S,U,cheap) sOffer(S,U,cheap)
sOffer(S,U,cheap) sOffer(S,U,exp)
sOffer(S,U,exp) sOffer(S,U,cheap)
sOffer(S,U,exp) sOffer(S,U,exp)

and so forth. This set determines what *V* knows with respect to its local view, its initial knowledge, and with respect to the actions and parameters we want to be confidential for *V*. What we require for this set is that for each sequence of actions in this set containing *sOffer(S,U,cheap)* at a specific point there must be another sequence of actions that contains *sOffer(S,U,exp)* at this specific point. This is the case in the above set of sequences of actions, thus parameter confidentiality with respect to all above listed conditions (local view, initial knowledge, relevant actions, possible parameters, etc.) is provided.

2.1.2.4.3 Confidentiality and privacy/anonymity

Privacy and anonymity as introduced in Section 2.1 reflect the property that the relation between an entity and a set of information is confidential. This can easily be expressed using the above concept of confidentiality: the parameter(s) to be confidential will be the entities performing specific actions, and/or other parameters of these actions expressing location or time. By specifying particular local views and initial knowledge for agents, and specific dependencies between actions with respect to knowledge about certain parameters, very fine-grained properties can be specified.

In Section 3.2 we will use an instantiation of the general concept of parameter confidentiality in order to formalize anonymity and privacy requirements. *Confidential(actions-to-learn-from, data-to-be-confidential, possible-values, allowed-dependencies, who)* denotes that no agents except those in the set *who* may learn the value of *data-to-be-confidential* from the actions in *actions-to-learn-from* although knowing the set *possible-values* that contains the possible values of the data to be confidential. In this first stage *allowed-dependencies* specifies the case in which no dependencies between actions are allowed to be known. This is the strongest requirement that can be specified and will probably be weakened once we know more about the architecture of the system. As already said before, concrete local views and

initial knowledge of agents can only be specified when the mechanisms to provide the security properties are identified.

2.2 Approach

The approach for deriving security requirements is based on a number of standards and best practice guidance documents [1][2][3][4]. The basic elements that are required of security and safety engineering processes are similar and include the following activities:

- develop a high-level (functional) model of the system to be analysed;
- identify safety hazards or security threats;
- classify the safety hazards or security risks;
- assess the associated risks;
- derive requirements for specific functions and assurance levels to mitigate the risks;
- evaluate the design and implementation for compliance with the requirements.

Although there are differences between safety and security engineering issues, there are also many similarities [10]. One of the aims in EVITA is to avoid a separation between security requirements and security issues with safety related implications.

The security engineering process for the EVITA project aims to infer security functional requirements based on the key methodology from ISO/IEC 15408 [1] and adopting the ISO/DIS 26262 [2] process together with systems engineering practices. The intentions of [1] are used to adapt the process for security issues and facilitate security evaluations. We emphasize, that we do not consider a complete security evaluation process according to [1] and highlight the differences to our process where necessary.

The methodology for inferring security functional requirements involves the following steps, which are based on the security requirements process described in [11]:

1. Description of system under investigation and its environment;
2. Description of relevant use cases (cf. Deliverable 2.1 [5]);
3. Identification of the assets to be protected within the described use cases (e.g. ECU, application/process, sensor, data, communication between system entities, etc.);
4. Identification of the threats posed to each asset in order to infer basic security functional requirements;
5. Evaluation and assignment of respective risks (probability of threat, cost/loss, risk classification);
6. Identification of respective security functional requirements for each threat according to risk analysis.

2.3 System under investigation and its environment

The system under investigation is an automotive on-board network consisting of embedded electronic control units (ECUs), sensors, and actuators that are connected with each other via some bus systems. Figure 1 shows the assumed on-board network architecture.

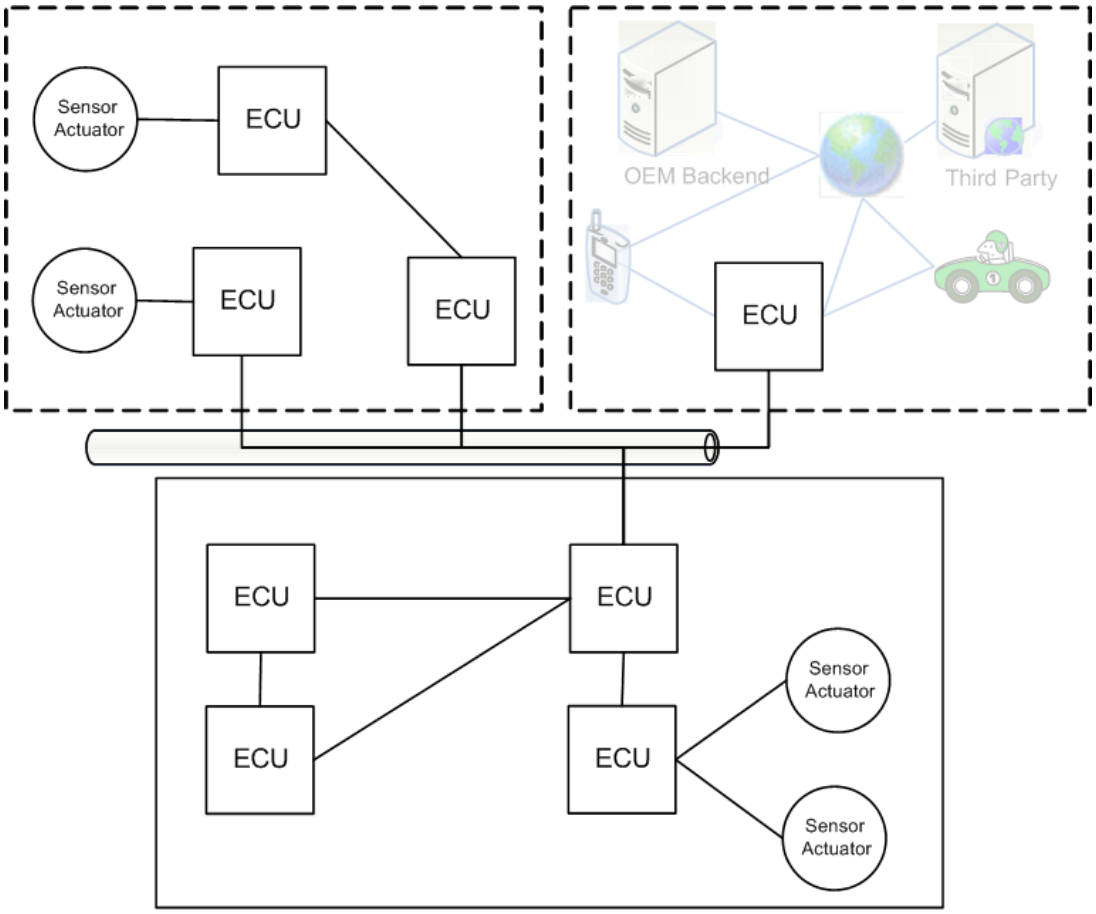


Figure 1 Generalised architecture of automotive on-board networks

The on-board network is assumed to possess interfaces to the outside for communicating with mobile devices, service providers, roadside units, and other vehicles:

- wireless interfaces such as GSM, UMTS, Bluetooth, W-LAN and DSRC and
- a wire-bound diagnostic interface.

For example, the on-board network may possess a Bluetooth interface in order to connect with mobile devices inside the car.

The generalised architecture of Figure 1 is too abstract for describing use cases. Therefore, use cases are described in [5] in terms of the reference architecture shown in Figure 2, which is an instantiation of the generalised architecture in Figure 1, based on recommendations originating from the EASIS project.

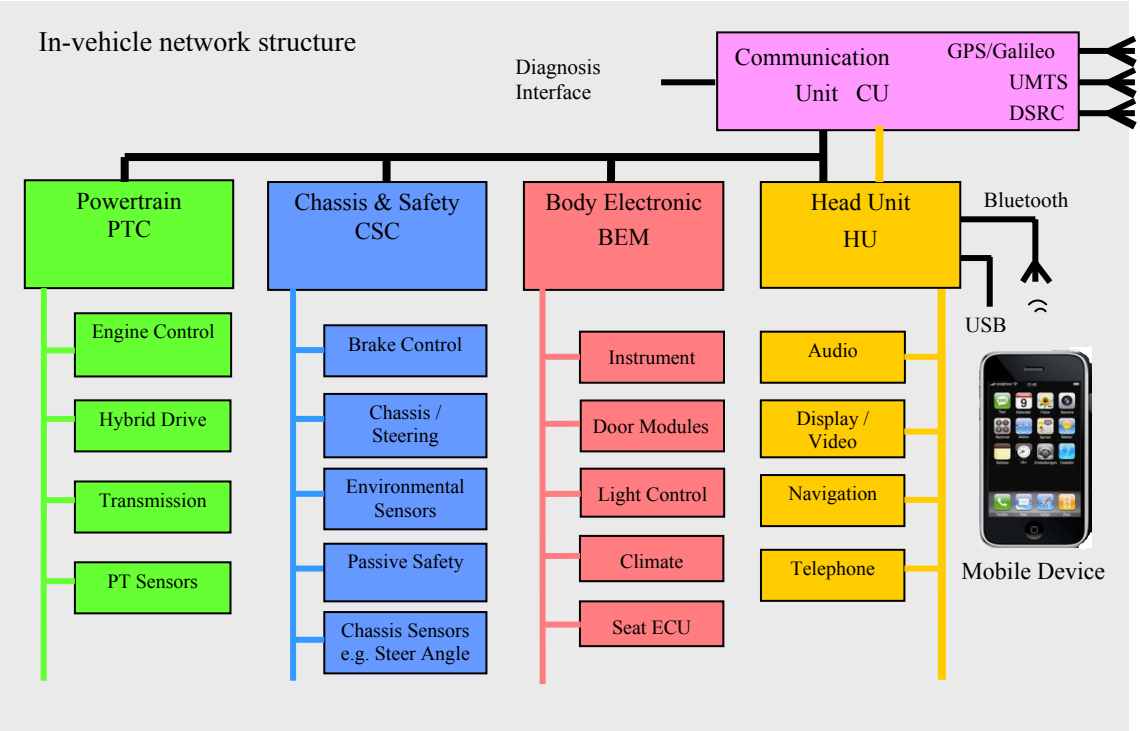


Figure 2 EVITA use case reference architecture

The system under investigation is assumed to operate in an uncontrolled environment. Therefore, the system under investigation must protect its assets against a variety of threats.

2.4 Summary of use cases

In order to identify requirements for systems it is necessary to have a conceptual model of how they should function. Use cases are an approach for building scenarios that describe the functional properties that may be required from a system in order to satisfy the goals of users.

The purpose of a use case is to describe the interaction between a system and the initiator of the interaction as a sequence of simple steps that are needed to achieve a specified goal. Use cases should focus on what the system must do, rather than how it is to be done, treating the system as a "black box". The interactions with the system, including system responses, should be described as perceived from outside the system.

For the purposes of the EVITA project, the use cases are intended to identify a range of specific future vehicle functions that could have possible security implications. The use cases are described in EVITA Deliverable D2.1 [5]. The development of these use cases involved the following steps:

- selection of a series of use cases appropriate to the objectives of EVITA;
- definition of the functionalities required to support the use cases;
- identification of relevant communication entities (e.g. vehicles, driver, backend infrastructure) and communication relations;
- specification of required data (in-vehicle, backend) as well as exchanged information;
- description of technical requirements (e.g. performance, bandwidth, distance, etc.) other than security.

The use cases are grouped into a number of categories for which e-security related intrusions were considered to be possible issues. The use cases that have been developed include the following:

- Car2MyCar (communication from other car to own car)

  Use case 1: Safety reaction: Active brake

  Use case 2: Local Danger Warning from other Cars

  Use case 3: Traffic Information from other Entities

- MyCar2Car (communication from own car to other car)

  Use case 4: Messages lead to safety reaction

  Use case 5: Local Danger Warning to other Cars

  Use case 6: Traffic Information to other Entities

- Car2I and I2Car (communication from car to infrastructure and from infrastructure to car)

  Use case 7: eTolling

  Use case 8: eCall

  Use case 9: Remote Car Control

  Use case 10: Point of Interest

- Nomadic Devices/USB Sticks/MP3

  Use case 11: Install applications

  Use case 12: Secure Integration

  Use case 13: Personalize the car

- Aftermarket

  Use case 14: Replacement of Engine ECU

  Use case 15: Installation of a Car2x Unit

- Workshop/Diagnosis

  Use case 16: Remote Diagnosis

  Use case 17: Remote Flashing

  Use case 18: Flashing per OBD

The use cases may themselves immediately suggest some security and safety requirements. However, they are also required as inputs to the “dark-side scenario” analysis, in order to identify the potential for malicious attacks. For example, interference with safety critical in-vehicular components and disruption of traffic flow by means of counterfeit messages are just two possibilities that need to be considered.

2.5 System assets

The main components of an automotive on-board network (see Figure 1 and Figure 2) that may become targets of attacks are:

- In-vehicle devices: ECUs, sensors and actuators,
- Safety critical and non-safety critical applications running on in-vehicle devices,
- Communication links internally within ECUs, between ECUs, between ECUs and sensors, between ECUs and actuators and between applications running on in-vehicle devices.

2.6 Threat identification (dark-side scenarios)

The purpose of developing the "dark-side" scenarios is to identify possible security threats and to allow aspects such as the desirability (to the attacker), opportunity, probability and severity of attacks to be assessed in order to support the security risk assessment activities.

The approach adopted in developing the dark-side scenarios for the EVITA project is based on the following elements:

- identification and classification of possible attack motivations;
- evaluation of associated attacker capabilities (e.g. technical, financial);
- attack modelling, comprising:
  - identification of specific attack goals that could satisfy the attack motivations;
  - construction of possible attack trees that could achieve attack goals, based on the functionality identified in the use cases [5].

This approach has already been used in the *Network-on-Wheels* project [12]. The attack trees are interpreted in terms of an initiating "attack goal", providing the attacker with an illegitimate benefit, which can be satisfied by one or more "attack objectives" that have a negative impact on the stakeholders. Each "attack objective" could be achieved by one or more attack methods, which may consist of one or more combinations of attacks on specific system assets.

Attacks that could have an impact on the safety of a car based on direct physical access in order to manipulate the hardware of that car (e.g. modification of ECUs or other electronic components) are excluded from this analysis as they are beyond the scope of the EVITA project. These classes of attacks are already feasible and probably always will be. While some outcomes of EVITA will help in the detection of malevolent modifications to a vehicle, this is not a specific objective of the project. Consequently, direct physical attacks against the hardware of the targets of attacks are out of scope. However, manipulations of devices that are under the control of the attacker are within the scope of EVITA (e.g. side channel attacks or extraction of keys); since attackers may modify their own vehicle in order to perform attacks against others.

Detailed results of the dark-side scenario analysis that are relevant to the security requirements analysis can be found in Appendix B – Dark-side scenarios.

2.7 Overview of risk analysis

In order to assess the "risk" associated with an attack it is necessary to assess the "severity" of the possible outcome for the stakeholders, and the "probability" that such an attack can be successfully mounted.

At the highest level, the security objectives are:

- **operational** – to maintain the intended operational performance of all vehicle and ITS functions;
- **safety** – to ensure the functional safety of the vehicle occupants and other road users;
- **privacy** – to protect the privacy of vehicle drivers, and the intellectual property of vehicle manufacturers and their suppliers;
- **financial** – to prevent fraudulent commercial transactions and theft of vehicles.

These security objectives counter generic security threats, as outlined in Table 1.

Table 1 Generic security threats and security objectives

Generic Security Threats				Security Objectives
Aims	**Target**	**Approach**	**Motivation**	
Harming individuals	Driver or passenger	Interference with safety functions of a specific vehicle	Criminal or terrorist activity	Safety Privacy
Harming groups	City or state economy, through vehicles and/or transport system	Interfere with safety functions of many vehicles or traffic management functions	Criminal or terrorist activity	Safety Operational
Gaining personal advantage	Driver or passenger	Theft of vehicle information or driver identity, vehicle theft, fraudulent commercial transactions	Criminal or terrorist activity	Privacy Financial
	Vehicle	Interference with operation of vehicle functions	Build hacker reputation	Operational Privacy
	Transport system, vehicle networks, tolling systems	Interference with operation of traffic management functions or tolling systems	Enhanced traffic privileges, toll avoidance,	Operational Privacy Financial
Gaining organizational advantage	Driver or passenger	Avoiding liability for accidents, vehicle or driver tracking	Fraud, criminal or terrorist activity, state surveillance	Privacy Financial
	Vehicle	Interference with operation of vehicle functions, acquiring vehicle design information	Industrial espionage or sabotage	Privacy Operational Safety

The severity of an attack is considered in terms of the 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 that are based on the severity classifications used in vehicle safety engineering. The severity of an attack is assessed using the attack trees, by considering the potential implications of the attack objectives for the stakeholders.

The probability of a successful attack is also derived from the attack trees, by identifying combinations of possible attacks on the system assets that could contribute to an attack method. Thus, the risk analysis is organized by attack tree, and decomposed down to asset

level. However, further decomposition may be helpful in estimating the probability of success (which is related to the "attack potential") for attacks on specific assets.

The probability and severity combinations are mapped to a series of risk levels ranging from 0 (lowest) to 6 (highest) in order to rank relative risks. In this scheme, high probability attacks with the severest outcomes have the highest risk levels, while low probability attacks with the least severe outcomes have the lowest risk levels. Between the extremes, the risk levels increase with rising probability and severity.

As severity is expressed in the form of a 4-component vector, the risk measure associated with an attack is also a 4-component vector. Furthermore, as several different attack methods may achieve the same attack objective, the result of the risk assessment is a set of risk vectors. This provides a convenient basis for systematically identifying threats that need to be countered with priority:

- Where a number of possible attack objectives may achieve the attack goal, the attack objective with the highest perceived risk level is the priority for countermeasures to reduce the risk level for the attack goal;
- Where a number of possible attack methods may lead to the same attack objective, the attack method with the highest perceived combined attack probability is the priority for countermeasures to reduce the risk level for the attack objective;
- Where a number of asset attacks may lead to the same attack method, the asset attack with the highest perceived attack probability (i.e. lowest attack potential) is the priority for countermeasures to reduce the risk level for the attack method.
- The repeated occurrence of particular attack patterns in attack trees is a further indicator for prioritising countermeasures that are likely to provide favourable cost-benefit properties.

A more detailed description of this process, including its application to the attack trees developed in Appendix B – Dark-side scenarios – can be found in Appendix C – Threat and risk analysis.

2.8 Identification of security requirements

2.8.1 Overview

Identification of security requirements in the EVITA project is based on two different but complementary viewpoints:

- **abstract functional path** – based on a purely functional representation of the use cases, providing security requirements by class (confidentiality, authenticity);
- **detailed functional path and mapping** – based on mapping a functional representation of the use cases to an architecture, providing both functional and architectural (availability, timing) requirements by use case

Merging the results of these two viewpoints should ensure that the security requirements are sufficiently comprehensive to support subsequent design activities. Brief overviews of these approaches are given below. More detailed descriptions can be found in Appendix D – Identifying security requirements.

2.8.2 Abstract functional path approach

2.8.2.1 Overview

The functional path model describes only the functional behaviour of the system under investigation and the information flows at its boundaries. Each information flow is associated with requirements for:

- establishing the ***authenticity*** of the incoming data and their origins;
- ensuring appropriate levels of ***confidentiality*** for the outgoing data.

This approach provides a very compact description of vehicle-to-X communications and a systematic approach to the identification of their associated security requirements.

2.8.3 Detailed functional path and mapping approach

2.8.3.1 Overview

The detailed functional path and mapping approach maps the functions to a generic architecture, allowing functional and architectural requirements to be identified. Consequently, aspects such as availability and timing, and dependencies between requirements, can be considered. An iterative process is employed, consisting of the following steps:

- Extract requirements from use cases
  - derive functional view
  - derive architectural mapping and correct functional view if necessary
- Verify coverage
  - attack trees
  - use case consistency/completeness
- Generate new requirements for unmatched threats / changed use cases
  - re-evaluate threat coverage

Additional benefits of this approach include more precise definition of the use cases, verification of existing attack trees, identification of new attacks, and more explicit mapping of security requirements to functions and assets.

A semi-formal description of security requirements based on SysML diagrams has several objectives:

- Describe security requirements with an approach close to the language of use case designers and with references to these use cases, based on both functional and mapping views. This specification distinctly aims at providing a system-oriented view, including timing and mapping issues, rather than an information-oriented view on requirements. We have been using the domains defined by use cases to classify the security requirements determined in addition to their security properties.
- Describe relationships between security requirements and in particular their respective dependencies.

- Describe in what way security requirements relate to attack trees (i.e., trace security requirements aimed at threat mitigation or anti-goal prevention rather than security property or goal achievement).
- One objective of this approach is to prove whether security requirement are met.
- Another objective is to determine which security mechanisms are to be defined in order to address or at least mitigate threats.

To identify security requirements, we have used the following methodology:

- For each use case, we derived one functional view of the system, using the UML composite structure diagrams defined in the DIPLODOCUS UML profile implemented in TTool [13]. TTool is an open-source toolkit supporting several UML2 profiles, including the DIPLODOCUS profile [20][21]. TTool has editing capabilities as well as simulation and formal verification capabilities. Indeed, all profiles implemented by TTool have a formal semantics defined as a translation to a process algebra. More specifically, the DIPLODOCUS profile targets the design space exploration of System-on-Chip. DIPLODOCUS stands for DesIgn sPace expLoration based on fOrmal Description teChniques, UML and SystemC. DIPLODOCUS follows the Y methodology which includes three views: functional view, architectural view, and then mapping view.
- For each use case, we also derived one mapping view of the system, using the UML deployment diagrams defined in the DIPLODOCUS UML profile implemented in TTool. That mapping view defines the locations where functions are executed. Functions are mapped either on hardware devices (sensors / actuators plus controller coming with those sensors/actuators) or on CPUs. For functions mapped on CPUs, we assume their code is stored, before execution, within the flash memory located on the same bus as the CPU. We also assume that, at execution time, the function code and data are stored within RAMs located on the same bus. Note that performing the mapping view has sometimes led to modifying the functional view directly derived from use cases, since one function can be mapped onto only one hardware execution node (i.e. at most on one CPU or one hardware device): when one function was to be mapped onto more than one hardware execution node, it was split into several sub-functions.
- Then, considering attack trees, use cases, functional and mapping views, we have settled on a list of security requirements. The latter have been modelled with the SysML diagrams implemented by TTool. The relations between requirements that have been considered are: *Containment*, *dependency* (<<*deriveReqt*>>), and reuse in different namespaces (<<*copy*>>). Those diagrams also contain observers, which may be seen as test cases meant to be used for the formal verification (or simulation) phase. Observers may additionally be seen as a means to document requirements. At last, a table of requirements is automatically derived from SysML diagrams. This set of requirements and observers altogether provides a conceptual model of the security expectations of the system, abstracted from the literary description of use cases.

2.8.3.2 Security requirements modelling

2.8.3.2.1 Overview

The security requirements defined using a SysML formalism have *"Requirement Containment Relationship"* and *"Derive Dependency"* SysML relationships [1]. A general methodology has been applied according to an iterative process, on functional and mapping views, on security requirements, and also on attack trees. The coverage and completeness of attack trees and use cases has thus been verified whilst listing security requirements.

Different semiformal definitions are used while defining security requirements, which will help us to prove future solutions.

2.8.3.2.2 Definition: Command

A command is an event or data (i.e. a message) sent from inside the on-board network to a function running on an actuator or on a sending device.

2.8.3.2.3 Definition: Functional Path

A use case always starts with a given message sent by one element outside of the on-board network to a function of the TOE. Let us call that message '*startMessage'*. We assume that '*startMessage'* is received by a function f_0. A use case is meant to produce commands. Let us call F_c the set of functions producing commands in the considered use case.

The functional path of a use case is a tuple consisting of a set C of events and data channels and of a set F of functions. C and F are defined as follows:

- *F_c is included into F*
- *C contains all channels which destination is a function of F*
- *F contains all functions that output messages in channels of C.*

Therefore, the functional path of a use case includes all data and events that are taken as an input by all functions involved in the direct or indirect production of commands defined in the use case.

Property: *The functional path of a use case is considered as valid if and only if, using the previous definition, f_0 is an element of F.*

3 EVITA Security Requirements

3.1 Security objectives

At the highest level, the security objectives (cf. Section 2.7) are:

- to maintain the intended operational performance of all vehicle and ITS functions;
- to ensure the functional safety of the vehicle occupants and other road users;
- to protect the privacy of vehicle drivers, and the intellectual property of vehicle manufacturers and their suppliers;
- to prevent fraudulent commercial transactions and theft of vehicles.

3.2 Security requirements

3.2.1 Overview

This section documents the security requirements that are needed to satisfy the stakeholders' security objectives considering the identified threats and assumed system architecture. Security requirements are constraints arising from security concerns; these requirements do not specify how the constraints are satisfied, but only what the constraint is. It is out of scope to address security mechanisms. The security requirements shall not make any assumptions regarding possible realisations. This is subject of the forthcoming task of secure on-board architecture specification.

The security requirements are based on the use cases [5] and attack trees (Appendix B – Dark-side scenarios) and derived in a systematic manner. The level of detail directly originates from the size of the use case model. The level of coverage is restricted to the amount of information that was input to the security analysis.

The fulfilment of security requirements is not measurable beyond Boolean (i.e. true or false). The fulfilment of security requirements in on-board architecture and protocol specifications will be verified by formal methods.

The following subsections list the security requirements determined using the two approaches outlined in the previous section, classified according to security properties. The requirements numbered under 100 correspond to requirements determined using the security-modelling framework SeMF, whereas requirements numbered above 100 were obtained following the system-oriented SysML approach. The latter are also described in a finer grained fashion, organized according to topics/diagrams, in Appendix D – Identifying security requirements.

3.2.2 Authenticity

Requirement reference: Authenticity_1
Informal description: Whenever an active braking action is performed, the own Environment Information measured by the sensors that the action is based on shall be authentic in terms of origin, content and time.
Semi-formal description: *auth(Environment-Sensing(car,Environment-Information,t),braking(car),Driver(car))*
Use case references: 1

Requirement reference: Authenticity_2
Informal description: Whenever an active braking action is performed, the own Vehicle Dynamics measured by the sensors that the action is based on shall be authentic in terms of origin, content and time.
Semi-formal description: *auth(Chassis-Sensing(car,Vehicle-Dynamics,t),braking(car),Driver(car))*
Use case references: 1

Requirement reference: Authenticity_3
Informal description: Whenever an active braking action is performed, the own Position-Information that the action is based on shall be authentic in terms of origin, content and time.
Semi-formal description: *auth(GPS-sensing(car,Position,t),braking(car),Driver(car))*
Use case references: 1

Requirement reference: Authenticity_4
Informal description: Whenever an active braking action is performed, the Position-Information of the original warning car that the action is based on shall be authentic in terms of origin, content and time.
Semi-formal description: *auth(GPS-Sensing(otherCar,Position,t),braking(car),Driver(car))*
Use case references: 1

Requirement reference: Authenticity_5
Informal description: Whenever an active braking action is performed, the sensor information of the original warning car that led to the warning and ultimately to the braking shall be authentic in terms of origin, content and time.
Semi-formal description: *auth(Chassis-Sensing(otherCar,Vehicle-Dynamics,t),braking(car),Driver(car))*
Use case references: 1, 4

Requirement reference: Authenticity_6
Informal description: Whenever an active braking action is performed, the position information for all vehicles that is being recorded in the neighbourhood tables shall be authentic for the braking action in terms of origin, content and time.
Semi-formal description: *auth(GPS-Sensing(allCars,Position,t),braking(car),Driver(car))*
Use case references: 1 (textual description only)
Notes: Analysis shows that this is an availability requirement, which appears here for the functional dependence of the braking

Requirement reference: Authenticity_7
Informal description: Whenever a Warning is shown on HMI, the own Position-Information that the action is based on shall be authentic in terms of origin, content and time.
Semi-formal description: *auth(GPS-Sensing(car,position,t),HMI-Display(car,Warning),Driver(car))*
Use case references: 2

Requirement reference: Authenticity_8
Informal description: Whenever a Danger-Warning is shown on HMI, the Position-Information of the warning car or the Cooperative-Awareness-Message sent by the RSU, depending on which the action is based on, shall be authentic in terms of origin, content and time.
Semi-formal description: *auth({GPS-Sensing(otherCar,position,t), send(RSU, CAM)}, HMI-Display(car,Warning), Driver(car))*
Use case references: 2, 5 (RSU from textual description)

Requirement reference: Authenticity_9
Informal description: Whenever a Danger-Warning is shown on HMI, the other vehicles sensor-information or the Co-operative-Awareness-Message sent by the RSU, depending on which the action is based on, shall be authentic in terms of origin, content and time.
Semi-formal description: *auth({sensing(otherCar, data,t), send(RSU, CAM)}, HMI-display(car,Warning), Driver(car))*
Use case references: 2, 5 (RSU from textual description)

Requirement reference: Authenticity_10
Informal description: Whenever a Navigation-Warning is shown on HMI, the own Position-Information that the action is based on shall be authentic in terms of origin, content and time.
Semi-formal description: *auth(GPS-Sensing(car,Position,t),HMI/Navigation-Display(Warning),Driver(car))*
Use case references: 3

Requirement reference: Authenticity_11
Informal description: Whenever a Navigation-Warning is shown on HMI, the Position-Information of the warning car or the Traffic-Information-Message sent by the RSU, depending on which the action is based on, shall be authentic in terms of origin, content and time.
Semi-formal description: *auth({GPS-Sensing(otherCar,position,t), send(RSU,TIM)}, HMI/Navigation-Display(Warning), Driver(car))*
Use case references: 3, 6

Requirement reference: Authenticity_12
Informal description: Whenever a Traffic Information Message is shown on HMI, the other vehicles' sensor-information or the Traffic Information Message sent by the RSU, depending on which the action is based on, shall be authentic in terms of origin, content and time.
Semi-formal description: *auth({sensing(otherCar, data,t), send(RSU, TIM)}, HMI/Navigation-Display(Warning),Driver(car))*
Use case references: 3, 6

Requirement reference: Authenticity_13
Informal description: Whenever a Traffic Information Message is received by the RSU, the sensor-data that it is based on shall be authentic in terms of origin, content and time.
Semi-formal description: *auth({sensing(otherCar, data,t), send(RSU, TIM)}, Processing-Showinfo,Driver(car))*
Use case references: 3, 6

Requirement reference: Authenticity_14
Informal description: Whenever eTolling information is received by the Service Provider's RSU, the accumulated position information at the car shall be authentic for the service provider.
Semi-formal description: *auth(GPS-Sensing(car,position,t),GSM-Receive(RSU(SP), car, Billing-Information),SP)*
Use case references: 7

Requirement reference: Authenticity_15
Informal description: Whenever an eCall request is received by the service provider, the position information that this was based on shall be authentic for the service provider in terms of origin, content and time.
Semi-formal description: *auth(GPS-Sensing(car,position,t),receive(SP,car,Crash-Info(Position)),SP)*
Use case references: 8

Requirement reference: Authenticity_16
Informal description: Whenever an eCall request is received by the service provider, the sensor information that this was based on shall be authentic for the service provider in terms of origin, content and time.
Semi-formal description: *auth(sensing(car,data,t),receive(SP,car,Crash-Info(Position)),SP)*
Use case references: 8

Requirement reference: Authenticity_17
Informal description: Whenever the car's hood is opened remotely, it shall be authentic for the owner that the command leading to this was sent by the allowed mobile device.
Semi-formal description: *auth(BT-Send(MobileDevice,openhood,t), open(car,hood), Owner(car))*
Use case references: 9

Requirement reference: Authenticity_18
Informal description: Whenever the PoI- (Point Of Interest) Information is displayed on the HMI, it shall be authentic for the driver that the information was sent by a PoI-Provider authorized by the driver.
Semi-formal description: *auth(send(PoI-Provider,PoI-Info),HMI-Show(car,PoI-Info),Driver(car))*
Use case references: 10

Requirement reference: Authenticity_19
Informal description: Whenever PoI (Point Of Interest) information is displayed on the HMI, it shall be authentic for the driver that this type of information is admitted by the PoI-Configuration (the driver's pre-configuration regarding the reception of PoI information).
Semi-formal description: *auth(HMI-Read(car,PoI-Configuration(PoI-Info-Type),t), HMI-Show(car,PoI-Info(PoI-Info-Type), Driver(car))*
Use case references: 10

Requirement reference: Authenticity_20
Informal description: Whenever a new software interface is displayed on the HMI, the software shall originate from an allowed mobile device.
Semi-formal description: *auth(USB-Receive(car,MobileDevice, Software),HMI-Show(SW-Interface),Driver(car))*
Use case references: 11

Requirement reference: Authenticity_21
Informal description: Whenever a new software interface is displayed on the HMI, the computed result shall be based on the user's inputs.
Semi-formal description: *auth(HMI-Read(car,driver(car), Inputs),HMI-Show(SW-Interface), Driver(car))*
Use case references: 11

Requirement reference: Authenticity_22
Informal description: Whenever a new external interface is displayed on the HMI, it shall originate from an allowed mobile device.
Semi-formal description: *auth(BT-Receive(car,MobileDevice,Display(Data)),HMI-Show(Data),Driver(car))*
Use case references: 12

Requirement reference: Authenticity_23
Informal description: Whenever inputs for an application installed on a mobile device are sent from the car to the mobile device, these inputs shall originate from the car's HMI.
Semi-formal description: *auth(HMI-Read(Inputs), BT-Send(car,MobileDevice,Inputs), Driver(car))*
Use case references: 12

Requirement reference: Authenticity_24
Informal description: Whenever the vehicle's seats are adjusted, it shall be authentic for the owner that the command leading to this was sent by an authorized mobile device.
Semi-formal description: *auth(BR-Receive(car,AuthDevice,SeatPosition),adjust(car,SeatPosition),Driver(car))*
Use case references: 13

Requirement reference: Authenticity_25
Informal description: Whenever an ECU is replaced in the car, it shall be authentically crafted by the manufacturer.
Semi-formal description: *auth(craft(Manufacturer,ECU),replace(car,ECU),Owner(car))* *auth(craft(Manufacturer,ECU),replace(car,ECU),Manufacturer(car))*
Use case references: 14
Notes: This property is related to a different system model, outside the runtime component-model of the car.

Requirement reference: Authenticity_26
Informal description: Whenever an ECU is added to the car, it shall be authentically crafted by the manufacturer.
Semi-formal description: *auth(craft(Manufacturer,ECU),install(car,ECU),Owner(car))* *auth(craft(Manufacturer,ECU),install(car,ECU),Manufacturer(car))*
Use case references: 15
Notes: This property is related to a different system model, outside the runtime component-model of the car.

Requirement reference: Authenticity_27
Informal description: Whenever diagnosis-data is sent to a maintenance-shop, it shall be authentic that it is a manufacturer-authorized maintenance-shop.
Semi-formal description: *auth(authorize(manufacturer,maintenance-shop),send(car,maintenance-shop,data),Owner(car))*
Use case references: 16

Requirement reference: Authenticity_28
Informal description: Whenever data is received by the maintenance-shop, it shall be authentic that it originates from the car.
Semi-formal description: *auth(store(car,data),receive(maintenance-shop,car,data),Owner(car))*
Use case references: 16
Notes: This is also an integrity requirement.

Requirement reference: Authenticity_29
Informal description: Whenever a firmware is installed to the car, it shall be authentically programmed by the manufacturer.
Semi-formal description: *auth(program(Manufacturer,Firmware),install(car,Firmware),Owner(car))* *auth(program(Manufacturer,Firmware),install(car,Firmware),Manufacturer(car))*
Use case references: 17, 18
Notes: This property is related to a different system model, outside the runtime component-model of the car.

Requirement reference: Authenticity_101
Informal description: Message source authentication along functional path: 1. Whenever a command (see Section 2.8.3.2.2) is sent from one internal ECU to another internal ECU, authentication of information along functional path must be ensured. **Use Case Reference**: 1, 2, 3, 4, 5, 6, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18 2. Whenever a message is received from a Mobile Device, authentication of all those messages must be ensured. **Use Case Reference**: 9, 11, 12, 13 3. Whenever a message is sent from vehicle to a Mobile Device, authentication of all those messages along functional path must be ensured. **Use Case Reference**: 9, 11, 12, 13
Semi-formal description: FSR-1.1.1 (General requirements – Fake Command related Requirements SysML diagram)

Requirement reference: Authenticity_102
Informal description: Code origin authentication: Whenever a command (see Section 2.8.3.2.2) is sent to ECU for flashing, code origin authentication must be ensured.
Use Case Reference: 14, 15, 17, 18
Semi-formal description: FSR-1.3.1 (General requirements –Fake Command related Requirements SysML diagram)

Requirement reference: Authenticity_103
Informal description: Authenticating message sources notifying a change in the environment coming in/out from vehicle: 1. Whenever an environment related information collected from gateways and sensors, data origin authentication of all those information must be ensured. **Use case Reference**: 1, 4, 5, 6. 2. Whenever an immediate danger message from environment sensors is received at Communication Unit (CU), data origin authentication of all messages must be ensured. **Use case Reference**: 1, 2, 4 3. Whenever a message (warning message) is received from other neighbourhood vehicles, data origin authentication of all messages along functional path must be ensured. **Use case Reference**: 1, 2, 3 4. Whenever CSC receives more information for plausibility check (vehicle dynamics data), data origin authentication of all information along functional path must be ensured. **Use case Reference**: 1, 2, 4, 5, 6, 5. Whenever a message (RSU, Traffic Light, Infrastructure based Server or other Vehicle) arrived at vehicle reception, data origin authentication of all those messages must be ensured. **Use Case Reference**: 1, 2, 3, 10, 16, 17, 18 6. Whenever a message is sent to RSU, data origin authentication of all those messages along functional path must be ensured. **Use Case Reference**: 7, 8, 16, 17 7. Whenever a message (warring message) is sent to other neighbourhood vehicles, data origin authentication of all those messages along functional path must be ensured. **Use Case Reference**: 4, 5, 6, 8. Whenever a warning is shown on the HMI, warning generated due to unexpected behaviour or warning message arrived from other vehicle, data origin authentication of these entire messages along functional path must be ensured. **Use Case Reference**: 2, 3, 5, 6, 9. Whenever an additional information messages (PoI) is received and showed, data origin authentication of these entire messages along functional path must be ensured. **Use Case Reference**: 10
Semi-formal description: GSR-1.3 (General requirements – Environment related Requirements SysML diagram)

3.2.3 Integrity

Unauthorized modification of data and functionality shall be prevented or at least detected. There shall be provisions for verifying the integrity of transported and stored data. This shall apply to both external and internal communications.

Requirement reference: Integrity_101
Informal description: Integrity of messages notifying a change in the environment coming in /out from vehicle: 1. Whenever an environment related information collected from gateways and sensors, Integrity of all those information must be ensured. **Use case Reference**: 1, 4, 5, 6. 2. Whenever an immediate danger message from environment sensors is received at Communication Unit (CU), Integrity of all messages must be ensured. **Use case Reference**: 1, 2, 4 3. Whenever a message (warning message) is received from other neighbourhood vehicles, Integrity of all messages along functional path must be ensured. **Use case Reference**: 1, 2, 3 4. Whenever CSC receives more information for plausibility check (vehicle dynamics data), Integrity of all information along functional path must be ensured. **Use case Reference**: 1, 2, 4, 5, 6, 5. Whenever a message (RSU, Traffic Light, Infrastructure based Server or other Vehicle) arrived at vehicle reception, Integrity of all those messages must be ensured. **Use Case Reference**: 1, 2, 3, 10, 16, 17, 18 6. Whenever a message is sent to a RSU, Integrity of all those messages along functional path must be ensured. **Use Case Reference**: 7, 8, 16, 17 7. Whenever a message (warning message) is sent to other neighbourhood vehicles, Integrity of all those messages along functional path must be ensured. **Use Case Reference**: 4, 5, 6, 8. Whenever a warning is shown on the HMI, warning generated due to unexpected behaviour, integrity of these entire messages along functional path must be ensured. **Use Case Reference**: 2, 3, 5, 6, 9. Whenever an additional information messages (PoI) is received and showed, integrity of these entire messages along functional path must be ensured. **Use Case Reference**: 10
Semi-formal description: GSR-1.1 (General Security Requirements – Environment related requirements SysML diagram)

Requirement reference: Integrity_102
Informal description: Flashing Command Integrity (the flashing went to its end): Whenever a flashing command (see Section 2.8.3.2.2) is sent to an ECU for flashing, integrity of flashing command must be ensured.
Use Case Reference:14, 15, 17, 18
Semi-formal description: **FBSR-4.1 (**Flashing per OBD use case specific requirements SysML diagram)

Requirement reference: Integrity_103
Informal description: It should be ensured that firmware data received as an update has not been modified since it left the manufacturer servers (code integrity): Whenever a flashing command (see Section 2.8.3.2.2) is sent to an ECU, the integrity of the firmware must be ensured.
Use Case Reference:14, 15, 17, 18
Semi-formal description: FSR-1.3.2 (General Security Requirements –Fake Command related requirements SysML diagram), FBSR-1.1.1 (Flashing per OBD use case specific requirements SysML diagram)

Requirement reference: Integrity_104
Informal description: Integrity of Message Attributes Along Functional Path. This particular requirement derives from the more general requirement of preventing Man-In-The-Middle attacks: 1. Whenever a command (see Section 2.8.3.2.2) is sent from one internal ECU to another internal ECU, integrity of information along functional path must be ensured. **Use Case Reference**: 1, 2, 3, 4, 5, 6, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18 2. Whenever a message is received from a Mobile Device, Integrity of all those messages must be ensured. **Use Case Reference**: 9, 11, 12, 13 3. Whenever a message is sent from vehicle to a Mobile Device, Integrity of all those messages along functional path must be ensured. **Use Case Reference**: 9, 11, 12, 13
Semi-formal description: FSR-1.1.3 (General requirements – Fake Command related Requirements SysML diagram)

Requirement reference: Integrity _105
Informal description: Ensure Correct Decision of Emergency Situation: Whenever an emergency situation happens and the driver or the vehicular system trigger to an emergency manoeuvre, the integrity of correcting decisions must be ensured along the functional path.
Use Case Reference: 1, 2, 4,5,8
Semi-formal description: BDOS-1.2-2.1 (Braking use case DoS Requirements SysML diagram)

3.2.4 Controlled access

Requirement reference: Access_101
Informal description: Controlled Access To Flashing Function: Whenever a flashing command (see Section 2.8.3.2.2) is sent to ECU, controlled access to flashing function must be ensured.
Use Case Reference: 14, 15, 17, 18
Semi-formal description: FBSR-1.1 (SysML General Requirements – Flashing Requirements)

Requirement reference: Access_102
Informal description: Controlled Access To Read From Flash: Whenever a flashing command (see Section 2.8.3.2.2) is sent to ECU, controlled access to read from flash must be ensured.
Use Case Reference: 14, 15, 17, 18
Semi-formal description: FBSR-1.2 (SysML General Requirements – Flashing Requirements)

3.2.5 Freshness

Requirement reference: Freshness_101

Informal description:

Freshness of messages carrying some environment related data and notifying a change in the environment coming in/out from vehicle should be ensured, in particular to prevent that replaying these data may trigger some undesirable behaviour from the TOE:

1. Whenever an environment related information is collected from gateways and sensors, freshness of all those information must be ensured.
   Use case Reference: 1, 4, 5, 6.
2. Whenever an immediate danger message from environment sensors is received at Communication Unit (CU), freshness of all messages must be ensured.
   Use case Reference: 1, 2, 4
3. Whenever a message (warning message) is received from other neighbourhood vehicles, freshness of all messages along functional path must be ensured.
   Use case Reference: 1, 2, 3
4. Whenever CSC receives more information for plausibility check (vehicle dynamics data), freshness of all information along functional path must be ensured.
   Use case Reference: 1, 2, 4, 5, 6,
5. Whenever a message (RSU, Traffic Light, Infrastructure based Server or other Vehicle) arrived at vehicle reception, freshness of all those messages must be ensured.
   Use Case Reference: 1, 2, 3, 10, 16, 17, 18
6. Whenever a message is sent to RSU, freshness of all those messages along functional path must be ensured.
   Use Case Reference: 7, 8, 16, 17
7. Whenever a message (warring message) is sent to other neighbourhood vehicles, freshness of all those messages along functional path must be ensured.
   Use Case Reference: 4, 5, 6,
8. Whenever a warning is shown on the HMI, warning generated due to unexpected behaviour or warning message arrived from other vehicle, freshness of these entire messages along functional path must be ensured.
   Use Case Reference: 2, 3, 5, 6,
9. Whenever an additional information messages (PoI) is received and showed, freshness of these entire messages along functional path must be ensured.
   Use Case Reference: 10

Semi-formal description:

GSR-1.2 (General requirements – Environment related Requirements SysML diagram)

Requirement reference: Freshness_102
Informal description: The freshness of the series of messages generated in sequence by all gateways or ECUs traversed along the functional path should be ensured in order to prevent the undesirable triggering of commands: 1. Whenever a message resulting in commands (see Section 2.8.3.2.2) is sent from one internal ECU to another internal ECU, freshness of information along functional path must be ensured. **Use Case Reference**: 1, 2, 3, 4, 5, 6, 8, 9, 10, 11, 12,13, 14, 15, 16, 17, 18 2. Whenever a message is received from a Mobile Device, freshness of all those messages must be ensured. **Use Case Reference**: 9, 11, 12, 13 3. Whenever a message is sent from vehicle to a Mobile Device, freshness of all those messages along functional path must be ensured. **Use Case Reference**: 9, 11, 12, 13
Semi-formal description: FSR-1.1.2 (General requirements – Fake Command related Requirements SysML diagram)

Requirement reference: Freshness_103
Informal description: Flashing command freshness: Whenever a command (see Section 2.8.3.2.2) is sent to ECU for flashing, flashing command freshness must be ensured.
Use Case Reference: 14, 15, 17, 18
Semi-formal description: FSR-1.3.3 (General requirements – Fake Command related Requirements SysML diagram)

3.2.6 Non-repudiation

The enforcement of non-repudiation is not necessary for the satisfaction of functional safety. Non-repudiation requirements arise when evidence of actions shall be presented to another entity later. Usually, these are motivated by legal requirements from law, liability or billing. Additional requirements might therefore arise during the forthcoming legal framework and requirements analysis.

Requirement reference: Proof-of-Authenticity_1
Informal description: The eTolling-Service Provider shall be able to prove the authenticity of the Billing-Information being based on the aggregated sensor data.
Semi-formal description: *non-rep-origin(GPS-Sensing(car,position,t),receive(SP,car, Billing-Information),SP)*
Use case references: 6
Notes: Compare with the requirement Authenticity_14.

3.2.7 Anonymity

Anonymity requirements target broadcast data-packets and require the identity to be confidential. In the semi-formal descriptions of these requirements *Lzero* describes the condition that no dependencies between actions are allowed to be known.

Requirement reference: Confidentiality_1
Informal description: The identity of the car shall be confidential. This includes especially those actions during which it is involved in wireless communication.
Semi-formal description: *confidential(actionsToLearnFrom1, car, allCars, Lzero, car)* *actionsToLearnFrom1={DSRC-Send(car, Neighbourhood-Token), DSRC-Send(car, C2X-Message(Emergency)), DSRC-Forward(car, C2X-Message(Emergency)), DSRC-Send(car, Cooperative-Awareness-Message), Send(car, Traffic-Information-Message), GSM-Send(car,Billing-Information), GSM-Send(car,eCall-Request), BT-Send(car,MobileDevice, InputData), BT-Send(MobileDevice,car,OpenHood), Send(MobileDevice, car, Software), BT-Send(MobileDevice,car,DisplayData), BT-Send(MobileDevice, car, SeatPosition), DSRC-Send(RSU(Manufacturer), car, Firmware)}*
Use case references: 1, 4, 5, 6, 7, 8, 9, 11, 12, 13, 17
Notes: This may be weakened by adjusting *Lzero* during the engineering to the use of pseudonymity.

Requirement reference: Confidentiality_2
Informal description: During operations that involve remote accesses to the vehicle (e.g. open hood, adjust seat, software install, integration etc.), the anonymity of the mobile device shall be guaranteed.
Semi-formal description: *confidential({BT-Send(MobileDevice,car,OpenHood), BT-Send(MobileDevice, car, SeatPosition), Send(MobileDevice, car, Software), BT-Send(MobileDevice, car, DisplayData), BT-Send(car,MobileDevice, Inputs)}, MobileDevice, allMobileDevices, Lzero, car)*
Use case references: 9, 11, 12, 13

3.2.8 Privacy

These requirements target every relation between identity and privacy-relevant information that are not already covered by the anonymity requirements.

Requirement reference: Confidentiality_3
Informal description: The position of a car at a certain point in time must be confidential.
Semi-formal description: *confidential({GPS-Sensing(car,Position), GSM-Send(car, Billing-Information)}, Position, allPositions, Lzero, car)*
Use case references: 7

Requirement reference: Confidentiality_4
Informal description: The personal information stored within the car shall remain confidential even during exchange of ECUs.
Semi-formal description: *confidential({ExchangeECU(Maintanance,car,ECU(Data)},Data, allData, Lzero, car)*
Use case references: 14

Requirement reference: Confidentiality_5
Informal description: The PoI-Configuration (the driver's preconfiguration regarding the reception of PoI- (Point of Interest) information) stored within the vehicle for a driver shall remain confidential even during exchange of data with an RSU.
Semi-formal description: *confidential({Receive(car,PoI-Info)}, PoI-Configuration, allPoIConfs, Lzero, car)*
Use case references: 10
Notes: The configuration of the vehicle can reveal personal information. In the case of PoIs, for example, personal preferences may be revealed.

Requirement reference: Confidentiality_6
Informal description: The seat position information for a driver shall remain confidential, even during exchange of data with a mobile device.
Semi-formal description: *confidential({Send(MobileDevice,car,SeatPosition)}, SeatPosition, allSeatPositions, Lzero, car)*
Use case references: 13
Notes: The configuration of the car can reveal personal information. In the case of SeatPosition, for example, the height of the driver could be inferred.

Requirement reference: Privacy_101
Informal description: Controlled access to e-service message data: an e-service message is a message sent from a car to an entity external to the TOE and car maker, and providing a service, for example: • service center residing in the infrastructure (e.g. eCall center, eToll) • garage (e.g. remote flashing) Whenever a message is sent from a vehicle to an entity external to the TOE and car maker, and providing a service, controlled access to e-service message data must be ensured.
Use Case Reference: 7, 8, 16, 17
Semi-formal description: PSR-1.1 (SysML General Requirements – Privacy Requirements)

Requirement reference: Privacy_102
Informal description: User Driven Privacy Policy: Users shall be able to determine by themselves the disclosure of information acceptable for various applications regarding their private profile or their car profile, providing it is lawful (e.g. car plates may need to be sent in some critical messages as required by law). That policy should be enforced according to user specifications. 1. Whenever a message is sent from a car to a RSU, PSAP and/or other fixed based server architecture, user driven privacy policy of all those messages must be ensured. **Use Case Reference**: 6, 7, 8 2. Whenever a message is sent from a car to car, user driven privacy policy of all those messages must be ensured. **Use Case Reference:** 5, 6
Semi-formal description: PSR1.2 (SysML General Requirements – Privacy Requirements)

Requirement reference: Privacy_103
Informal description: Car2X message anonymity: The sending of a critical message should not make it possible to connect the driver to other messages previously sent, e.g. promiscuous listening of the network 1. Whenever a message is sent from a car to a RSU, PSAP and/or other fixed based Server architecture, anonymity of all those messages must be ensured. **Use Case Reference**: 7, 8 2. Whenever a message is sent from a car to car, anonymity of all those messages must be ensured. **Use Case Reference**: 5
Semi-formal description: PSR-1.3.1 (SysML General Requirements – Privacy Requirements)

Requirement reference: Privacy_104
Informal description: Unlinkable driver identification between services: some applications will need to prevent two different services from linking their respective knowledge of the drivers
Semi-formal description: PSR-1.3.2 (SysML General Requirements – Privacy Requirements)

Requirement reference: Privacy_105
Informal description: Unlinkable time ordering of messages: some applications will need to prevent or limit the possibility to order a set of predetermined critical messages in order to gain indirect information about the behaviour of a driver. 1. Whenever a message is sent from a car to a RSU, PSAP and/or other fixed based Server architecture, unlinkable time ordering of all those messages must be ensured. **Use Case Reference**: 6, 7, 8 2. Whenever a message is sent from a car to car, unlinkable time ordering of all those messages must be ensured. **Use Case Reference**: 5, 6
Semi-formal description: PSR1.3.3 (SysML General Requirements – Privacy Requirements)

3.2.9 Confidentiality

These requirements target classic confidentiality of transferred bilateral data.

Requirement reference: Confidentiality_7
Informal description: The Billing-Information shall remain confidential between the car and the RSU.
Semi-formal description: *confidential({GSM-Send(car,Billing-Information), Billing-Information, allBillingInfos, Lzero, {car, RSU(SP)})*
Use case references: 7
Notes: The confidentiality of the billing agent is already captured by Confidentiality_1.

Requirement reference: Confidentiality_101
Informal description: Firmware data should remain confidential when updates are distributed by the manufacturer: Whenever a flashing command (see Section 2.8.3.2.2) is sent to ECU, confidentiality of firmware data must be ensured.
Use Case Reference: 14, 15, 17, 18
Semi-formal description: FBSR-1.2.1 (SysML General requirements – Flashing Requirements)

Requirement reference: Confidentiality_102
Informal description: Confidentiality of firmware update should be ensured: attackers should not gain information out of the flashing process about the version of firmware being installed or the ECU being updated: Whenever a flashing command (see Section 2.8.3.2.2) is sent to ECU, confidentiality of firmware update must be ensured.
Use Case Reference: 14, 15, 17, 18
Semi-formal description: FBSR-1.2.1.1 (SysML General requirements – Flashing Requirements)

3.2.10 Availability

Requirement reference: Availability_101
Informal description: The availability of the bus should be ensured for some applications (especially safety critical ones): Whenever information is exchanged between different ECU's, CU, HU, Sensors, and other units of vehicle availability of Bus must be ensured.
Use Case Reference: 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18. (some of them could be considered with lower priority: 9, 10, 11, 12, 13, 14, 15)
Semi-formal description: ASR-1.1 (SysML General Requirements – Availability Requirements)

Requirement reference: Availability_102
Informal description: The availability of ECU CPUs should be ensured for some applications (especially those that require some computation or message routing to take place): Whenever information is exchanged between different ECU's, CU, HU, Sensors, and other units of vehicle availability of CPU must be ensured.
Use Case Reference: 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18. (some of them could be considered with lower priority: 9, 10, 11, 12, 13, 14, 15)
Semi-formal description: ASR-1.2 (SysML General Requirements – Availability & Overhead Requirements), ASR-1.2 (SysML General Requirements – Availability Requirements)

Requirement reference: Availability_103
Informal description: The availability of RAM attached to an ECU should be ensured (to access some the ECU software or some data): Whenever information is exchanged between different ECU's, CU, HU, Sensors, and other units of vehicle availability of RAM must be ensured.
Use Case Reference: 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18. (some of them could be considered with lower priority: 9, 10, 11, 12, 13, 14, 15)
Semi-formal description: ASR-1.3 (SysML General Requirements – Availability Requirements)

Requirement reference: Availability_104

Informal description:

The availability of external communication device should be ensured for applications that need to communicate with the environment of the TOE:

1. Whenever information is sent from vehicle to neighbourhood vehicles, RSU, or others entities, availability of external communication device (communication unit) must be ensured.
   Use Case Reference: 1, 4, 5, 6, 7, 8, 11, 12, 15, 16, 17, 18 (some of them could be considered with lower priority: 11, 12, 15)
2. Whenever information is received for a vehicle from neighbourhood vehicles, RSU, or other authorized entities, availability of external communication device (communication unit) must be ensured.
   Use Case Reference: 1, 2, 3, 7, 8, 10, 11, 12, 15, 16, 17, 18 (some of them could be considered with lower priority: 10, 11, 12, 15)

Semi-formal description:

ASR-1.4 (SysML General Requirements – Availability Requirements)

Requirement reference: Availability_105

Informal description:

The availability of the radio medium should be ensured for applications that need to communicate with the environment of the TOE:

1. Whenever information is sent from vehicle to neighbourhood vehicles, RSU or other entities, availability of radio medium (antennas) must be ensured.
   Use Case Reference: 1, 4, 5, 6, 7, 8, 9, 11, 12, 13, 14, 15, 16, 17 (some of them could be considered with lower priority: 9, 11, 12, 13, 15)
2. Whenever information is received for a vehicle from neighbourhood vehicles, RSU or other authorized entities, availability of radio medium (antennas) must be ensured.
   Use Case Reference: 1, 2, 3, 7, 8, 9, 10, 11, 12, 13, 15, 16, 17 (some of them could be considered with lower priority: 9, 10, 11, 12, 13, 15)

Semi-formal description:

ASR-2 (SysML General Requirements – Availability Requirements)

Requirement reference: Availability_106

Informal description:

The highest availability should be ensured for highest priority functions (in particular those essential to safety-critical applications):

1. Whenever information is sent from vehicle to neighbourhood vehicles, RSU or others entities highest availability of requested devices must be ensured for highest priority functions.
   Use Case Reference: 1, 4, 5, 6, 7, 8, 16, 17, 18
2. Whenever information is received for a vehicle from neighbourhood vehicles, RSU or other entities, highest availability of requested devices must be ensured for highest priority functions.
   Use Case Reference: 1, 2, 3, 7, 8, 16, 17, 18

Semi-formal description:

ASR-3 (SysML General Requirements – Availability Requirements)

Requirement reference: Availability_107
Informal description: Prevent Broadcast Brake DoS When Emergency Situation: Whenever an emergency brake message was broadcasted to a vehicle, availability of broadcasting system must be ensured.
Use Case Reference: 5
Semi-formal description: BDOS-2.2 (Braking use case DoS Requirements SysML diagram)

Requirement reference: Availability_108
Informal description: Brake Total Response Time: Whenever an emergency brake was triggered by the driver or automatically generated by the vehicular system availability of the braking total response time must be ensured.
Use Case Reference: 1, 4, 5, 8
Semi-formal description: BDOS-1.1 (Braking use case DoS Requirements SysML diagram)

3.3 Priority of security requirements

Analysis of the attack trees demonstrates that specific asset attacks may contribute to different attack objectives within the same attack tree, and may also contribute to attack objectives associated with other attack trees. For a particular asset attack, both the risk level (which reflects the severity of outcome for an attack, and the attack potential associated with the asset attacks that contribute to it) and the number of instances from the collection of attack trees are indicators of the importance of the asset attack and the likely benefits of countermeasures for reducing the probability of successful attacks of this nature.

The severity measure is considered in terms of a four-component vector that reflects potential safety, operational, privacy and financial aspects that may be associated with a security attack (see Section C.1.2). For safety-related security threats the "controllability" of the hazard by the driver (see [2][3]) constitutes an additional dimension for the probability contribution to the relative risk level. The severity, controllability and attack potential estimates relating to the asset attacks identified from the attack trees are detailed in Appendix C. The proposed mapping of these parameters to relative risk level is summarised in Table 2, where non-safety risks and highly controllable safety-related risks are associated with controllability C=1, and only safety–related risks are associated with the higher controllability measures (i.e. Table 2 combines Table 9 and Table 11 of Appendix C). In principle the relative risk is also a four-component vector, inheriting this property from the severity, although in the EVITA analysis it is usually found to be of lower order. The class "R7+" that is used in Table 2 denotes levels of risk that are unlikely to be considered acceptable, such as safety hazards with the highest severity classes and threat levels, coupled with very low levels of controllability.

Table 2 Combined risk graph for safety-related (C≥1) and non-safety (C=1) security threats

Controllability (C)	Severity (S_i)	Combined Attack Probability (A)				
		A=1	A=2	A=3	A=4	A=5
C=1	S_i=1	R0	R0	R1	R2	R3
	S_i=2	R0	R1	R2	R3	R4
	S_i=3	R1	R2	R3	R4	R5
	S_i=4	R2	R3	R4	R5	R6
C=2	S_S=1	R0	R1	R2	R3	R4
	S_S=2	R1	R2	R3	R4	R5
	S_S=3	R2	R3	R4	R5	R6
	S_S=4	R3	R4	R5	R6	R7
C=3	S_S=1	R1	R2	R3	R4	R5
	S_S=2	R2	R3	R4	R5	R6
	S_S=3	R3	R4	R5	R6	R7
	S_S=4	R4	R5	R6	R7	R7+
C=4	S_S=1	R2	R3	R4	R5	R6
	S_S=2	R3	R4	R5	R6	R7
	S_S=3	R4	R5	R6	R7	R7+
	S_S=4	R5	R6	R7	R7+	R7+

Analysis of the attack trees, which were based on the EVITA use cases [5] and an assumed architecture based on the EASIS project, has identified small numbers of possible attack methods on various system assets that could lead to the achievement of potential attacker objectives. These “asset attacks” represent the terminal nodes of the attack trees, and specific subsets of the security requirements that are considered to be necessary to protect against such attacks have been identified. The risk analysis identifies severity at the higher levels of the attack trees and works up associated probability measures from the asset attacks that terminate the lower levels of the attack trees. Thus, the attack trees, risk analysis and security requirements are mapped to each other via the concept of asset attacks.

The same asset attacks often appear in more than one of the attack trees, but may be associated with different risk levels because the severity measures differ between trees. The results of the EVITA risk analysis activity (detailed in Appendix C – Threat and risk analysis) are therefore summarized in terms of the number of occurrences of particular risk levels associated with specific asset attacks in Table 3, which also lists the security requirements to counter each such asset attack. Thus, Table 3 also provides an indication of the relative importance of the security requirements detailed in Section 3.2.

The risk level reported in Table 3 is based on the worst case where more than one element of the risk vector is present in the risk analysis tables. Where alternative attack routes are available, the associated risk level is adjusted to reflect the attack probability for the asset attacks involved. Consequently, Table 3 indicates the worst case risk estimates for all of the attack alternatives listed in the risk analysis tables. Thus, if high risk asset attacks are mitigated by appropriate security countermeasures, the only change required to Table 3 is to remove or modify the entries corresponding to the risks that have been mitigated. The risk levels associated with lower risk attack alternatives remain unchanged.

Table 3 Summary findings of risk analysis

Identified threats		**Risk analysis results**		**Security requirements**
Asset	***Attack***	***Risk level***	***Number of instances***	
Chassis Safety Controller	Denial of service	1 2	3 1	Authenticity_6, Availability_102, Availability_106
	Exploit implementation flaws	4 5	1 1	Authenticity_1, Authenticity_2, Authenticity_3, Authenticity_4, Authenticity_5, Authenticity_101, Authenticity_102, Authenticity_103, Confidentiality_101, Confidentiality_102, Integrity_101, Integrity_102, Integrity_103, Integrity_104, Integrity_105, Freshness_101, Freshness_102, Freshness_103, Availability_101, Availability_102, Availability_103, Availability_106, Availability_107, Availability_108, Privacy_101, Privacy_102, Privacy_103, Privacy_104, Privacy_105
	Corrupt data or code	3	1	Authenticity_1, Authenticity_2, Authenticity_5, Authenticity_6, Authenticity_101, Authenticity_102, Authenticity_103, Confidentiality_101, Confidentiality_102, Integrity_101, Integrity_102, Integrity_103, Integrity_104, Integrity_105, Freshness_101, Freshness_102, Freshness_103, Availability_101, Availability_102, Availability_103, Availability_106, Availability_107, Availability_108, Access_101, Access_102, Privacy_101, Privacy_102, Privacy_103, Privacy_104, Privacy_105
	Flash malicious code	4 5 6	1 1 1	Authenticity_1, Authenticity_2, Authenticity_3, Authenticity_4, Authenticity_5, Authenticity_101, Authenticity_102, Authenticity_103, Confidentiality_101, Confidentiality_102, Integrity_101, Integrity_102, Integrity_103, Integrity_104, Integrity_105, Freshness_101, Freshness_102, Freshness_103, Availability_101, Availability_102, Availability_103, Availability_106, Availability_107, Availability_108, Access_101, Access_102, Privacy_101, Privacy_102, Privacy_103, Privacy_104, Privacy_105

Identified threats		Risk analysis results		Security requirements
Asset	***Attack***	***Risk level***	***Number of instances***	
Wireless Communications	Corrupt or fake messages	2 3 4 5 6 7	5 5 4 1 4 3	Authenticity_4, Authenticity_6, Authenticity_8, Authenticity_9, Authenticity_11, Authenticity_12, Authenticity_13, Authenticity_14, Authenticity_15, Authenticity_16, Authenticity_17, Authenticity_18, Authenticity_20,Authenticity_22, Authenticity_23, Authenticity_24, Authenticity_27, Authenticity_28, Confidentiality_1, Confidentiality_2, Authenticity_101, Authenticity_102, Authenticity_103, Integrity_101, Integrity_102, Integrity_103, Integrity_104, Integrity_105, Availability_102, Availability_106, Availability_107, Availability_108, Privacy_101, Privacy_103
	Jamming	4 5	3 2	Availability_103, Availability_104, Availability_105, Availability_107, Availability_108, Integrity_102
	Listen, intercept, alter, inject, replay	2 3 4 5	2 11 2 1	Authenticity_4, Authenticity_6, Authenticity_8, Authenticity_9, Authenticity_11, Authenticity_12, Authenticity_13, Authenticity_14, Authenticity_15, Authenticity_16, Authenticity_17, Authenticity_18, Authenticity_20, Authenticity_22, Authenticity_23, Authenticity_24, Authenticity_27, Authenticity_28, Confidentiality_1, Confidentiality_2, Confidentiality_3, Authenticity_101, Authenticity_102, Authenticity_103, Confidentiality_101, Confidentiality_102, Integrity_101, Integrity_102, Integrity_103, Integrity_104, Integrity_105, Freshness_101, Freshness_102, Freshness_103, Availability_102, Availability_105, Availability_107, Availability_108, Privacy_101, Privacy_103, Privacy_104, Privacy_105
	Exploit vulnerability or implementation error	2 3	2 3	Authenticity_4, Authenticity_6, Authenticity_8, Authenticity_9, Authenticity_11, Authenticity_12, Authenticity_13,Authenticity_14, Authenticity_15, Authenticity_16, Authenticity_17, Authenticity_18, Authenticity_20, Authenticity_22, Authenticity_23, Authenticity_24, Authenticity_27, Authenticity_28, Confidentiality_1, Confidentiality_2, Confidentiality_3, Authenticity_101, Authenticity_102, Authenticity_103, Confidentiality_101, Confidentiality_102, Integrity_101, Integrity_102, Integrity_103, Integrity_104, Integrity_105, Freshness_101, Freshness_102, Freshness_103, Availability_102, Availability_105, Availability_107, Availability_108, Privacy_101, Privacy_103, Privacy_105

Identified threats		**Risk analysis results**		**Security requirements**
Asset	***Attack***	***Risk level***	***Number of instances***	
In-car communications	Jamming	3	4	Availability_101, Availability_103, Availability_104, Availability_105, Availability_107, Availability_108, Availability_106, Integrity_102
	Insert fake data	1	1	Authenticity_1, Authenticity_2, Authenticity_3, Authenticity_4, Authenticity_5, Authenticity_6, Authenticity_7, Authenticity_8, Authenticity_9, Authenticity_10, Authenticity_11, Authenticity_12, Authenticity_13, Authenticity_14, Authenticity_15, Authenticity_16, Authenticity_17, Authenticity_18, Authenticity_24, Authenticity_27, Authenticity_28, Authenticity_101, Authenticity_103, Availability_102, Availability_106, Privacy_101, Privacy_103, Privacy_105, Integrity_105
	Disable or denial of service	4	1	Authenticity_1, Authenticity_2, Authenticity_3, Authenticity_4, Authenticity_5, Authenticity_6, Authenticity_7, Authenticity_8, Authenticity_9, Authenticity_10, Authenticity_11, Authenticity_12, Authenticity_13, Authenticity_14, Authenticity_15, Authenticity_16, Authenticity_17, Authenticity_18, Authenticity_24, Authenticity_27, Authenticity_28, Availability_101, Availability_103, Availability_104, Availability_105, Availability_107, Availability_108, Availability_106, Integrity_102
	Listen, intercept, alter, inject, replay	2 3 4	6 2 3	Authenticity_1, Authenticity_2, Authenticity_3, Authenticity_4, Authenticity_5, Authenticity_6, Authenticity_7, Authenticity_8, Authenticity_9, Authenticity_10, Authenticity_11, Authenticity_12, Authenticity_13, Authenticity_14, Authenticity_15, Authenticity_16, Authenticity_17, Authenticity_18, Authenticity_24, Authenticity_27, Authenticity_28, Confidentiality_3, Authenticity_101, Authenticity_102, Authenticity_103, Confidentiality_101, Confidentiality_102, Integrity_101, Integrity_102, Integrity_103, Integrity_104, Integrity_105, Freshness_101, Freshness_102, Freshness_103, Availability_101, Availability_102, Availability_103, Availability_104, Availability_107, Availability_108, Availability_106, Access_101, Access_102, Privacy_101, Privacy_102, Privacy_103, Privacy_105
	Configuration change	2 3	1 1	Integrity_101, Integrity_102, Integrity_103, Integrity_104

Identified threats		**Risk analysis results**		**Security requirements**
Asset	***Attack***	***Risk level***	***Number of instances***	
	Exploit vulnerability or implementation error	2 3	1 1	Authenticity_1, Authenticity_2, Authenticity_3, Authenticity_4, Authenticity_5, Authenticity_6, Authenticity_7, Authenticity_8, Authenticity_9, Authenticity_10, Authenticity_11, Authenticity_12, Authenticity_14, Authenticity_14, Authenticity_15, Authenticity_16, Authenticity_17, Authenticity_18, Authenticity_24, Authenticity_27, Authenticity_28, Confidentiality_3, Authenticity_101, Authenticity_102, Authenticity_103, Confidentiality_101, Confidentiality_102, Integrity_101, Integrity_102, Integrity_103, Integrity_104, Integrity_105, Freshness_101, Freshness_102, Freshness_103, Availability_101, Availability_102, Availability_103, Availability_104, Availability_105, Availability_106, Availability_107, Availability_108, Privacy_101, Privacy_101, Privacy_102, Privacy_103, Privacy_105
GPS	Jamming	5	1	Authenticity_3, Authenticity_4, Authenticity_7, Authenticity_8, Authenticity_10, Authenticity_11, Authenticity_14, Authenticity_15, Availability_106, Integrity_102
	Spoofing	3 4 5 6 7	4 3 2 1 1	Authenticity_3, Authenticity_4, Authenticity_7, Authenticity_8, Authenticity_10, Authenticity_11, Authenticity_14, Authenticity_15, Authenticity_103, Availability_106, Privacy_101, Privacy_103, Privacy_105
Communications Unit	Denial of service	3 4	3 2	Authenticity_3, Authenticity_4, Authenticity_6, Authenticity_7, Authenticity_8, Authenticity_9, Authenticity_10, Authenticity_11, Authenticity_12, Authenticity_13, Authenticity_14, Authenticity_15, Authenticity_16, Authenticity_17, Authenticity_27, Authenticity_28, Availability_102, Availability_104, Availability_106, Availability_107, Availability_108, Integrity_102

Identified threats		**Risk analysis results**		**Security requirements**
Asset	***Attack***	***Risk level***	***Number of instances***	
	Corrupt code or data	2	1	Authenticity_3, Authenticity_4, Authenticity_6, Authenticity_7, Authenticity_8, Authenticity_9, Authenticity_10, Authenticity_11, Authenticity_12, Authenticity_13, Authenticity_14, Authenticity_15, Authenticity_16, Authenticity_17, Authenticity_27, Authenticity_28, Confidentiality_1, Confidentiality_3, Confidentiality_7, Authenticity_101, Authenticity_102, Authenticity_103, Confidentiality_102, Integrity_101, Integrity_102, Integrity_103, Integrity_104, Integrity_105, Freshness_101, Freshness_102, Freshness_103, Availability_101, Availability_102, Availability_103, Availability_104, Availability_105, Availability_106, Availability_107, Availability_108, Access_101, Access_102, Privacy_101, Privacy_102, Privacy_103, Privacy_104, Privacy_105
	Exploit vulnerability of external communication protocols	2 3	4 2	Authenticity_3, Authenticity_4, Authenticity_6, Authenticity_8, Authenticity_9, Authenticity_11, Authenticity_12, Authenticity_13, Authenticity_14, Authenticity_15, Authenticity_16, Authenticity_17, Authenticity_27, Authenticity_28, Confidentiality_1, Confidentiality_3, Confidentiality_7, Authenticity_101, Authenticity_102, Authenticity_103, Confidentiality_102, Integrity_101, Integrity_102, Integrity_103, Integrity_104, Integrity_105, Freshness_101, Freshness_102, Freshness_103, Availability_101, Availability_102, Availability_104, Availability_105, Availability_106, Availability_107, Availability_108, Privacy_101, Privacy_102, Privacy_103, Privacy_104, Privacy_105
	Malware flashed with OBD update	1 2 3 5 6	3 2 1 1 1	Authenticity_3, Authenticity_4, Authenticity_6, Authenticity_8, Authenticity_10, Authenticity_11, Authenticity_12, Authenticity_13, Authenticity_14, Authenticity_15, Authenticity_16, Authenticity_17, Authenticity_27, Authenticity_28, Confidentiality_1, Confidentiality_3, Confidentiality_7, Authenticity_101, Authenticity_102, Authenticity_103, Confidentiality_102, Integrity_101, Integrity_102, Integrity_103, Integrity_104, Integrity_105, Freshness_101, Freshness_102, Freshness_103, Availability_101, Availability_102, Availability_103, Availability_104, Availability_105, Availability_106, Availability_107, Availability_108, Access_101, Access_102, Privacy_101, Privacy_102, Privacy_103, Privacy_104, Privacy_105

Identified threats		**Risk analysis results**		**Security requirements**
Asset	***Attack***	***Risk level***	***Number of instances***	
	Malware delivered by mobile device	1 2 3	1 2 1	Authenticity_3, Authenticity_4, Authenticity_6, Authenticity_8, Authenticity_9, Authenticity_10, Authenticity_11, Authenticity_12, Authenticity_13, Authenticity_14, Authenticity_15, Authenticity_16, Authenticity_27, Authenticity_28, Confidentiality_1, Confidentiality_3, Confidentiality_7, Authenticity_101, Authenticity_102, Authenticity_103, Confidentiality_102, Integrity_101, Integrity_102, Integrity_103, Integrity_104, Integrity_105, Freshness_101, Freshness_102, Freshness_103, Availability_101, Availability_102, Availability_103, Availability_104, Availability_105, Availability_106, Availability_107, Availability_108, Access_101, Access_102, Privacy_101, Privacy_102, Privacy_103, Privacy_104, Privacy_105
In-car sensors	Manipulate	4 7	4 1	Authenticity_1, Authenticity_2, Authenticity_5, Authenticity_9, Authenticity_12, Authenticity_13, Authenticity_16, Authenticity_103, Availability_106
	Malware flashed	3 4	1 1	Authenticity_1, Authenticity_2, Authenticity_5, Authenticity_9, Authenticity_12, Authenticity_13, Authenticity_16, Authenticity_101, Authenticity_102, Authenticity_103, Confidentiality_101, Confidentiality_102, Integrity_101, Integrity_102, Integrity_103, Integrity_104, Integrity_105, Freshness_101, Freshness_102, Freshness_103, Availability_101, Availability_102, Availability_103, Availability_106, Availability_107, Availability_108, Access_101, Access_102, Privacy_101, Privacy_102, Privacy_103, Privacy_104, Privacy_105
	Disable or denial of service	4	1	Authenticity_1, Authenticity_2, Authenticity_5, Authenticity_9, Authenticity_12, Authenticity_13, Authenticity_16, Availability_102, Availability_106, Availability_107, Availability_108, Integrity_102
	Spoof	3 4	2 1	Authenticity_1, Authenticity_2, Authenticity_5, Authenticity_9, Authenticity_12, Authenticity_13, Authenticity_16, Authenticity_103, Availability_106, Access_101, Access_102, Privacy_101, Privacy_103, Privacy_104, Privacy_105

Identified threats		**Risk analysis results**		**Security requirements**
Asset	***Attack***	***Risk level***	***Number of instances***	
Head unit	Gain root access to embedded OS	1 3 4	1 2 2	Authenticity_7, Authenticity_8, Authenticity_9, Authenticity_10, Authenticity_11, Authenticity_12, Authenticity_17, Authenticity_18, Authenticity_19, Authenticity_20, Authenticity_21, Authenticity_22, Authenticity_23, Confidentiality_1, Confidentiality_2, Authenticity_101, Authenticity_102, Authenticity_103, Confidentiality_101, Confidentiality_102, Integrity_101, Integrity_102, Integrity_103, Integrity_104, Integrity_105, Freshness_101, Freshness_102, Freshness_103, Availability_101, Availability_102, Availability_103, Availability_105, Availability_106, Availability_107, Availability_108, Access_101, Access_102, Privacy_101, Privacy_102, Privacy_103, Privacy_104, Privacy_105
	Exploit vul-nerability or implementa-tion error	2 3 4	2 2 1	Authenticity_7, Authenticity_8, Authenticity_9, Authenticity_10, Authenticity_11, Authenticity_12, Authenticity_17, Authenticity_18, Authenticity_19, Authenticity_20, Authenticity_21, Authenticity_22, Authenticity_23, Confidentiality_1, Confidentiality_2, Authenticity_101, Authenticity_102, Authenticity_103, Confidentiality_101, Confidentiality_102, Integrity_101, Integrity_102, Integrity_103, Integrity_104, Integrity_105, Freshness_101, Freshness_102, Freshness_103, Availability_101, Availability_102, Availability_103, Availability_105, Availability_106, Availability_107, Availability_108, Access_101, Access_102, Privacy_101, Privacy_102, Privacy_103, Privacy_104, Privacy_105
	Malware flashed	1 5	1 1	Authenticity_7, Authenticity_8, Authenticity_9, Authenticity_10, Authenticity_11, Authenticity_12, Authenticity_17, Authenticity_18, Authenticity_19, Authenticity_20, Authenticity_21, Authenticity_22, Authenticity_23, Confidentiality_1, Confidentiality_2, Authenticity_101, Authenticity_102, Authenticity_103, Confidentiality_101, Confidentiality_102, Integrity_101, Integrity_102, Integrity_103, Integrity_104, Integrity_105, Freshness_101, Freshness_102, Freshness_103, Availability_101, Availability_102, Availability_103, Availability_105, Availability_106, Availability_107, Availability_108, Access_101, Access_102, Privacy_101, Privacy_102, Privacy_103, Privacy_104, Privacy_105
In-car ECU	Disable or denial of service	2	1	Authenticity_25, Authenticity_26, Availability_102, Availability_106, Availability_107, Availability_108, Integrity_102

Identified threats		Risk analysis results		Security requirements
Asset	***Attack***	***Risk level***	***Number of instances***	
	Malware flashed with OBD update	1 2	4 3	Authenticity_25, Authenticity_26, Confidentiality_4, Confidentiality_5, Confidentiality_6, Authenticity_101, Authenticity_102, Authenticity_103, Confidentiality_101, Confidentiality_102, Integrity_101, Integrity_102, Integrity_103, Integrity_104, Integrity_105, Freshness_101, Freshness_102, Freshness_103, Availability_101, Availability_102, Availability_103, Availability_106, Availability_107, Availability_108, Access_101, Access_102, Privacy_101, Privacy_102, Privacy_103, Privacy_104, Privacy_105
	Configura-tion change	2 3	1 1	Authenticity_25, Authenticity_26, Authenticity_29, Confidentiality_4, Confidentiality_5, Confidentiality_6, Integrity_101, Integrity_102, Integrity_103, Integrity_104, Integrity_105, Availability_106
	Exploit vul-nerability or implementa-tion error	2 3	1 1	Authenticity_25, Authenticity_26, Confidentiality_4, Confidentiality_5, Confidentiality_6, Authenticity_101, Authenticity_102, Authenticity_103, Confidentiality_101, Confidentiality_102, Integrity_101, Integrity_102, Integrity_103, Integrity_104, Integrity_105, Freshness_101, Freshness_102, Freshness_103, Availability_101, Availability_102, Availability_103, Availability_106, Availability_107, Availability_108, Access_101, Access_102, Privacy_101, Privacy_102, Privacy_103, Privacy_104, Privacy_105
Powertrain Controller	Malware flashed with OBD update	1 5 6	2 1 1	Authenticity_101, Authenticity_102, Authenticity_103, Confidentiality_101, Confidentiality_102, Integrity_101, Integrity_102, Integrity_103, Integrity_104, Integrity_105, Freshness_101, Freshness_102, Freshness_103, Availability_101, Availability_102, Availability_103, Availability_106, Availability_107, Availability_108, Access_101, Access_102, Privacy_101, Privacy_102, Privacy_103, Privacy_104, Privacy_105
Powertrain Peripherals	Corrupt code or data	1	2	Authenticity_101, Authenticity_102, Authenticity_103, Confidentiality_101, Confidentiality_102, Integrity_101, Integrity_102, Integrity_103, Integrity_104, Integrity_105, Freshness_101, Freshness_102, Freshness_103, Availability_102, Availability_103, Availability_106, Availability_107, Availability_108, Access_101, Access_102, Privacy_101, Privacy_102, Privacy_103, Privacy_104, Privacy_105

Identified threats		Risk analysis results		Security requirements
Asset	***Attack***	***Risk level***	***Number of instances***	
In-car Interfaces	Physical access	3	1	Authenticity_101, Authenticity_102, Authenticity_103, Integrity_101, Integrity_102, Integrity_103, Integrity_104, Integrity_105, Freshness_101, Freshness_102, Freshness_103, Availability_101, Availability_102, Availability_103, Availability_104, Availability_106, Availability_107, Availability_108, Access_101, Access_102, Privacy_101, Privacy_103, Privacy_105
	Exploit vulnerabilities, introduce bogus data	3 4 5	1 1 1	Authenticity_17, Authenticity_24, Confidentiality_5, Confidentiality_6, Confidentiality_1,
Roadside Unit	Exploit configuration errors	2 3	2 3	Authenticity_8, Authenticity_9, Authenticity_11, Authenticity_12, Authenticity_13, Authenticity_14, Authenticity_18, Confidentiality_1, Confidentiality_7, Confidentiality_1, Authenticity_101, Authenticity_103, Integrity_101, Integrity_102, Integrity_103, Integrity_104, Integrity_105, Freshness_101, Freshness_102, Freshness_103, Availability_105, Availability_107, Availability_108, Access_101
	Gain root access	2 3	1 1	Authenticity_8, Authenticity_9, Authenticity_11, Authenticity_12, Authenticity_13, Authenticity_14, Authenticity_18, Confidentiality_1, Confidentiality_7, Confidentiality_1, Authenticity_101, Authenticity_103, Integrity_101, Integrity_102, Integrity_103, Integrity_104, Integrity_105, Freshness_101, Freshness_102, Freshness_103, Availability_102, Availability_105, Availability_107, Availability_108, Access_101, Privacy_101, Privacy_103, Privacy_104, Privacy_105
	Exploit protocol implementation flaws	2 3 5	3 3	Authenticity_8, Authenticity_9, Authenticity_11, Authenticity_12, Authenticity_13, Authenticity_14, Authenticity_18, Confidentiality_1, Confidentiality_7, Authenticity_101, Authenticity_103, Confidentiality_102, Integrity_101, Integrity_102, Integrity_103, Integrity_104, Integrity_105, Freshness_101, Freshness_102, Freshness_103, Availability_102, Availability_105, Availability_107, Availability_108, Access_101, Privacy_101, Privacy_103, Privacy_104, Privacy_105

Identified threats		**Risk analysis results**		**Security requirements**
Asset	***Attack***	***Risk level***	***Number of instances***	
	Physical access		*Not in scope*	Authenticity_8, Authenticity_9, Authenticity_11, Authenticity_12, Authenticity_13, Authenticity_14, Authenticity_18, Confidentiality_1, Confidentiality_7, Authenticity_101, Authenticity_103, Integrity_101, Integrity_102, Integrity_103, Integrity_104, Integrity_105, Freshness_101, Freshness_102, Freshness_103, Availability_102, Availability_105, Availability_107, Availability_108, Access_101, Privacy_101, Privacy_103, Privacy_104, Privacy_105
Roadside Unit to Authority Communications	Listen, intercept, alter, inject, replay		*Not in scope*	Authenticity_101, Authenticity_103, Integrity_101, Integrity_102, Integrity_103, Integrity_104, Integrity_105, Freshness_101, Freshness_102, Freshness_103, Availability_102, Access_101, Privacy_101, Privacy_103, Privacy_104, Privacy_105
E-call Service Centre Interfaces	Denial of service	1	3	Availability_105, Availability_106
	Exploit interfaces	3	3	Authenticity_15, Authenticity_16, Confidentiality_1, Authenticity_101, Authenticity_103, Availability_102, Availability_106
E-call Service Centre	Overload	2	3	Availability_106
Keys	Illegal acquisition, modification or breaking	1 2	1 3	Authenticity_25, Authenticity_26, Authenticity_29, Confidentiality_4, Confidentiality_5, Confidentiality_6, Authenticity_101, Authenticity_102, Authenticity_103, Confidentiality_101, Confidentiality_102, Integrity_101, Integrity_102, Integrity_103, Integrity_104, Integrity_105, Freshness_101, Freshness_102, Freshness_103, Availability_103, Availability_106, Availability_107, Availability_108, Access_101, Access_102, Privacy_101, Privacy_102, Privacy_103, Privacy_104, Privacy_105

4 Conclusions

Future automotive safety applications based on vehicle-to-vehicle and vehicle-to-infrastructure communication entail new security requirements for automotive on-board networks. Security measures that are intended to counter potential threats will inevitably contribute to the cost and complexity of future systems. Consequently, there is a need to ensure that such measures are commensurate with the perceived risks. This report endeavours to put forward a process to determine which security measures are appropriate and cost effective onto an objective basis. The security engineering process is applied to an exemplary automotive on-board network, considering exemplary use cases and typical threats.

The security requirements that were derived from this process are based on analysis of a range of possible applications, rather than a specific and well-defined system, and assume an underlying architectural topology. These requirements will require further refinement in order to develop specifications for more concrete system development, where design decisions will affect the need for and implementation of measures to respond to these requirements. This process will be further investigated within the EVITA project in the secure on-board architecture design and verification.

Appendix A – Glossary

This appendix provides a list of key terms and their definitions as well as the source of these definitions. Cross-references within this glossary are indicated by the use of bold italics in the definitions.

Where possible, the definitions used are taken directly from relevant standards or other publications. In a number of cases, terms and phrases are used with a particular meaning in this document for which explicit definitions have not been identified in relevant standards or other publications. In these cases the source of these definitions is indicated as "EVITA". In some cases the definitions are cross-referenced to sections within this document that contain more detailed descriptions. In some cases a definition from a relevant standard has been adapted for the purposes of this work, in which case both the EVITA version and the standard version (in brackets) are presented together in order to illustrate the differences.

Term	Definition	Source
anonymity	the property that the relation between an entity and its identity is *confidential* to authorized entities	Section 2.1.1.7
assets	information or resources that could be subject to *attack* (possibly, but not necessarily, requiring protection)	EVITA
	[entities that the owner of the *TOE* presumably places value upon]	ISO/IEC 15408-1 Rev. 3 (draft)
	[information or resources to be protected by the countermeasures of a *TOE*]	ISO/IEC 15408-1
asset attack	a particular type of *attack* on a specific *asset*	EVITA
assurance	grounds for confidence that an entity meets its security objectives.	ISO/IEC 15408-1
attack	exploitation of vulnerabilities to obtain unauthorized access to or control of *assets*	EVITA
attacker	an individual or group aiming to mount an *attack*	EVITA
attack goal	the ultimate objective of an *attack*, providing the *attacker* with a benefit of some kind through achieving a degree of *harm* to one or more *stakeholders*	EVITA
attack method	one or more possible combinations of *asset attacks* that could achieve a specific *attack objective*	EVITA

attack objective	one or more system states or conditions affecting the stakeholders that could satisfy a particular *attack goal*	EVITA
attack potential	measure of the effort to be expended in attacking a *TOE*, expressed in terms of an attacker's expertise, resources and motivation	ISO/IEC 15408-1 Rev. 3 (draft)
	[the perceived potential for success of an *attack*, should an attack be launched, expressed in terms of an *attacker*'s expertise, resources and motivation]	ISO/IEC 15408-1
attack probability	qualitative measure of the likelihood of a successful *attack* using a numerical scale mapped to the five *attack potential* classes of the Common Criteria, ranging from 1 (corresponding to "beyond high") up to 5 (corresponding to "basic")	EVITA
attack tree	graphical representation of possible sequences of events to implement an attack, derived from an initiating *attack goal*	B. Schneier, "Secrets and Lies", Chapter 21, Wiley, 2000 [16]
attacker type	classification of *attacker* by budgetary resources, technical skills, and motivation	EVITA
automotive safety integrity level (ASIL)	one of four classes to specify the item's necessary *safety requirements* for achieving an acceptable *residual risk* with D representing the highest and A the lowest class	ISO/DIS 26262-1
authenticity	includes *data origin authenticity* as well as *integrity* (in terms of format and content) and *freshness* for specified information	Section 2.1.2.3
availability	the property that the specified device or service is operational when required	Section 2.1.1.9
confidentiality	the property that specified information can only be accessed by authorized entities	Section 2.1.1.8
controllability	avoidance of a specified *harm* or damage through timely reactions of the persons involved	ISO/DIS 26262-1
controlled access	the property that only authorized entities can access specified data or perform specified actions	Section 2.1.1.4
combined attack probability	estimated *attack probability* for an *attack method* involving a particular combination of *asset attacks*	EVITA

data origin authenticity	the property that specified information truly originates from the claimed source	Section 2.1.1.2
evaluation	assessment of a *PP*, an *ST* or a *TOE*, against defined criteria.	ISO/IEC 15408-1
evaluation assurance level (EAL)	a package consisting of assurance components from ISO/IEC 15408-3 that represents a point on ISO/IEC 15408 predefined assurance scale	ISO/IEC 15408-1
exposure	state of being in an operational situation that may be *hazardous* if coincident with the *attack method* under consideration	EVITA
	[state of being in an operational situation that may be *hazardous* if coincident with the failure mode under consideration]	ISO/DIS 26262-1
formal	expressed in a restricted syntax language with defined semantics based on well-established mathematical concepts	ISO/IEC 15408-1
formal verification	mathematical proof of an algorithm or a specification against properties	ISO/DIS 26262-1
freshness	the property that specified information is not a copy of the same information received by an entity at an earlier time	Section 2.1.1.5
harm	a negative impact on *stakeholders* (in terms of physical safety, privacy, personal or organisational finances, or operational performance) either directly or indirectly as a result of *attacks* on vehicle systems or their operating environment	EVITA
	[physical injury or damage to the health of people either directly or indirectly as a result of damage to property or the environment]	ISO/DIS 26262-1
hazard	potential source of *harm*	ISO/DIS 26262-1
hazardous event	coincidence of *hazard* and *exposure*	ISO/DIS 26262-1
informal	expressed in natural language	ISO/IEC 15408-1
informal notation	description technique that does not have its syntax completely defined	ISO/DIS 26262-1
integrity	the property that specified information remains unchanged between observations	Section 2.1.1.3
non-repudiation	The property that an entity cannot deny that an action was performed by that entity	Section 2.1.1.6

organisational security policies	one or more security rules, procedures, practices, or guidelines imposed by an organisation upon its operations	ISO/IEC 15408-1
privacy	the property that the relation between an entity and specified information is *confidential* to authorized entities	Section 2.1.1.7
protection profile (PP)	an implementation-independent set of security requirements for a category of *TOE*s that meet specific consumer needs.	ISO/IEC 15408-1
residual risk	*risk* remaining after protective measures have been taken	MISRA Safety Analysis, 2007
risk	combination of the probability of occurrence of *harm* and the *severity* of that harm	ISO/DIS 26262-1
risk graph	mapping of combinations of qualitative *severity* and probability measures associated with possible *harm* to a qualitative *risk level* scale	EVITA
risk level	qualitative ranking of the relative *risk* associated with possible *harm* based on a range of qualitative *severity* and probability measures	EVITA
safety functions	functions to be implemented by a *safety-related system*, which are intended to maintain a safe state in respect of specified *hazards*	MISRA Safety Analysis, 2007
safety integrity	the degree of confidence in a *safety-related system* satisfactorily performing the required *safety functions* under all the stated conditions within a stated period of time	MISRA Safety Analysis, 2007
safety integrity level (SIL)	discrete level for specifying the *safety integrity* requirements of the *safety functions* allocated to *safety-related systems*, where SIL 4 has the highest level of safety integrity and SIL 1 has the lowest	MISRA Safety Analysis, 2007
safety-related system	a designated system that: • implements the required *safety functions* necessary to achieve or maintain a safe state for the total system; and • is intended to achieve, on its own or with other systems, the necessary *safety integrity* for the required *safety functions*	MISRA Safety Analysis, 2007

safety requirements	the requirements of the *safety functions* that have to be fulfilled by the *safety-related systems*	MISRA Safety Analysis, 2007
security function (SF)	a part or parts of the *TOE* that have to be relied upon for enforcing a closely related subset of the rules from the *TSP*	ISO/IEC 15408-1
security target (ST)	a set of security requirements and specifications to be used as the basis for evaluation of an identified *TOE*	ISO/IEC 15408-1
semiformal	expressed in a restricted syntax language with defined semantics	ISO/IEC 15408-1
severity	measure of the expected degree of *harm* to *stakeholders* associated with a specific *attack objective*	EVITA
	[measure of the expected degree of *harm* to an endangered individual in a specific situation]	ISO/DIS 26262-1
stakeholders	individuals and/or organizations that may suffer *harm* as a result of a successful *attack* on one or more *assets* (may include vehicle users, other road users, ITS service operators, civil authorities, vehicle manufacturers and system suppliers)	EVITA
target of evaluation (TOE)	an IT product or system and its associated guidance documentation that is the subject of an *evaluation*	ISO/IEC 15408-1
TOE security policy (TSP)	a set of rules that regulate how *assets* are managed, protected and distributed within a *TOE*	
verification	determination of completeness and correctness of specification or implementation of requirements from a previous phase	ISO/DIS 26262-1

Appendix B – Dark-side scenarios

B.1 Introduction

The main objectives of the dark-side scenario analysis are, firstly, to identify security threats and, secondly, to provide a basis for assessing risk, which reflects the severity and probability of attacks. We use the approach of attack trees [16] for this purpose. Attack trees are related to fault trees, which are normally used for identifying safety hazards.

In order to support risk analysis, we propose to structure the attack trees in the following manner. The root of an attack tree (Level 0) is an abstract "attack goal" that is associated with a benefit to the attacker of some kind. Its child nodes (Level 1) represent different "attack objectives" that could satisfy this attack goal. The attack objectives have a negative impact on the stakeholders (e.g. vehicle users, other road users, ITS service operators, civil authorities, vehicle manufacturers and system suppliers). Thus, the severity of the outcome can be estimated at this level. The attack objectives may be further decomposed into a number of "attack methods" that could be employed to achieve the attack objective. Each attack method will in turn be based on a logical combination (AND/OR) of attacks against one or more "assets" populating the lowest levels of the attack tree. These are described here as "asset attacks", and are the terminal nodes of the tree. The tree is truncated where the probability of success can be estimated for asset attacks. These individual probabilities can subsequently be combined using the tree logic to assess the overall probability for each of the attack methods.

This generic tree structure is illustrated in Figure 3. The possible depth of the analysis is inevitably more limited in the early concept stage than when specific design and implementation decisions have been made.

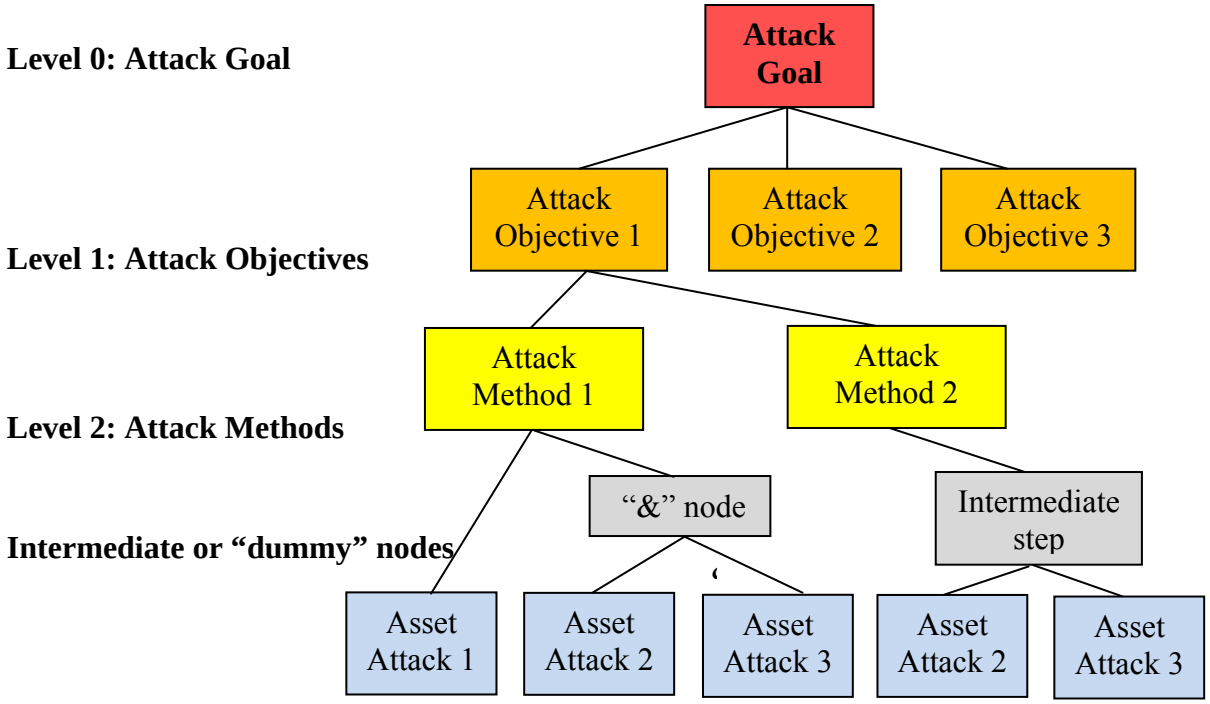


Figure 3 Generic attack tree structure

B.2 Attack motivations

B.2.1 Overview

Possible attack motivations can be broadly categorized as follows:

- harming an individual
  - driver or passenger
  - for the purposes of criminal or terrorist activity
- harming groups
  - e.g. drivers, city/state economy
  - for the purposes of criminal or terrorist activity
- gaining personal advantage
  - e.g. identity or information theft, vehicle theft, fraudulent commercial transactions, enhanced traffic privileges
  - for the purposes criminal activity
- gaining organisational advantage
  - e.g. avoiding liability for accidents, acquiring vehicle design information
  - for the purposes of fraud, industrial/state espionage or sabotage

The attack motivations suggest particular types of attackers and attacker capabilities, as well as associated attack goals.

B.2.2 Do psychological or physical harm to the driver

The goal of the attacker is to harm the driver. While this global goal can somehow be related to others, like gaining financial or personal advantages, we consider only the situations where there is no other aim in the attack. Harming the driver (and / or the car occupants) can be refined in several subclasses:

- Undermining the reputation of the driver (either her self-esteem or from a legal point of view). In order to compromise the reputation, the attacker will very likely impersonate her or her car in one way or another and perform actions with these stolen identities, e.g. violating some laws (speed limits) while pretending to be the victim.
- By preventing her from using her vehicle (denial of use). The denial of use attack intends to cause some damage, like missing an important date. Remote control of the target vehicle is an interesting possibility because more direct actions involving physical access may be considered as more dangerous by the attacker.
- By injuring her (or worse). While not necessarily the most important attack, it is probably the first one that comes in mind when dealing with automated cars because it is most spectacular. The attacker will probably try to cause an accident.

B.2.3 Gain information about the driver

In general the growing number of sensor allows measuring many quantities. The data from these measurements are processed and partly stored. The attacker will either try to intercept the communication between the sensors and the rest of the ICT system or get access to the memory where this data is stored. Car data allows the construction of a profile for the car – the time of activity, location, braking, acceleration and steering. This information can be used for different purposes – lawful as well as un-lawful ones – that all have a negative impact on the car owner or driver's privacy:

- Law enforcement authorities will try to get access to mobility profiles in order to identify cars that have been involved in accident and criminal activities.
- Insurance companies have a high interest in getting access to car data in order to calculate a premium based on the individual risk related to the way a car is driven.
- Criminals could use the information from the mobility profile and actual data about the car's location for planning to steal or hijack the car or to kidnap the driver.

B.2.4 Gain reputation as a hacker

The main (maybe the only) goal of the attacker is to gain reputation by breaking/hacking the system and publishing the results afterward. The publication of the results or merely the fact that the system has been broken/hacked is the main goal – otherwise no reputation is gained. Since the publication of the attack is the main goal no real harm is done through this attack. Real harm will be only caused if the attacker finds a design flaw that is very hard to fix or maybe even un-fixable or if they have secondary goals like: financial gain.

Other goals could result in reputation as a hacker: Hacking/reverse engineering in order to create homebrew applications (e.g. for the Head-Unit). The attacker's goal will not be reputation but rather the ability to run his own software on parts of the system installed in his car.

B.2.5 Financial gain

A financial gain is probably the motive behind most attacks. There are several possible combinations of attackers and motivations to break into the system of a car:

- First, after an accident the car owner could try to manipulate the data stored in the vehicle to obscure culpable behaviour like exceeding the speed limit or driving with too little distance to the car in front. For this purpose it would also be possible to impersonate another vehicle. In certain cases the attacker could also manipulate the vehicle software and pretend that it was not up to date in order to make the vendor liable fort he damage.
- Second, a third party or criminals could tamper with the vehicle for new types of insurance fraud: by causing another vehicle to brake or steer they could provoke an accident in order to get a high compensation from the insurance company. Another possibility is, when the car system is used to authenticate the driver/car for the utilisation of a charged service (ranging from parking fees and tolls to charged entertainment content and software downloads) an attacker can try to steal the driver or car's identity and impersonate as this car/driver. Another way would be to increase the regular usage costs of the vehicle by

increasing its fuel consumption, by damaging some expensive parts, by reducing the time interval between services, etc.

- Finally, experience also shows that any new device that is integrated in the Internet will become subject of spamming. Attackers may easily find ways to send spam to mobile ICT devices that are used in the car but they will certainly aim to place their messages on displays that demand the driver's attention.

B.2.6 Gain personal advantages (non financial)

Personal advantages can be gained in different ways and for different purposes. One example is to attack road regulations in order to go faster through the traffic or to stop other vehicles in the traffic. Possible methods could be to force a green wave, e.g. getting all traffic lights in front of the attacker to switch to green. Another one is to manipulate the traffic flow by directing other cars to alternative routes or clearing any traffic jam in front of the attacker. Finally a way of gaining a personal advantage would consist in manipulating the speed limits. In particular, it might be possible to tamper with the infrastructure so that other vehicles are notified of a lower speed limit than reality.

Another purpose could be to gain access to special areas like secured parking lots, fair areas or similar, which is controlled by the vehicle identity. A possible method is to impersonate the ID of authorized car or person, which was gained before.

B.2.7 Gain information about vehicle manufacturer

While most of the described attacks aimed at the drivers and their direct environment another motivation can be to attack the car manufacturer. There are several reasons possible. First attackers can try to steal intellectual property of the manufacturer by accessing to the vehicles software, e.g. another manufacturer aims to disclose technical specs and to imitate them. Methods that could be used to achieve this target could be reverse engineering methods unauthorised diagnosis; mobility and status profiles; us probing to extract crypto material; get unencrypted firmware from flashes

Another way of attack the vehicle manufacturer can be destroying his reputation. This could be done in several ways, for example by manipulating the safety of a car to harm random owners of one car manufacturer's cars. Another example could be the disclosure or compromise of privacy to destroy reputation. While these attacks aim at destroying the public reputation other attacks could aim at financial harm for the manufacturer. Possible targets in this could be to reduce the life expectancy of a car or damage a car by manipulating the engine control. Another possibility is to provoke unexpected behaviour or switch off of car functions. Finally manipulations of the power train actuator could lead to higher fuel consumption or reduce of service intervals could damage the reputation and end in expensive lawsuits against a manufacturer.

B.2.8 Harm the economy

This attack and underlying objectives should be envisaged at an organizational scale. It makes use of potential attacks on the car platform to disrupt the economical value of the car-related business by wreaking havoc to the road infrastructure.

This attack consists in the large-scale manipulation of traffic in order to generate huge traffic jams, therefore rendering driving virtually impossible. This attack might in particular make use of a variety of techniques used more directly for other attacks, simply in order to disrupt the normal service of roads. Two approaches in particular represent avenues for large scale attacks. First, protocols with the infrastructure might be subject to attacks with respect to car-related information in order to tamper with signalling. In particular impersonation attacks simulating the presence of emergency vehicles are quite likely. Such attacks might in particular result in signal lights being turned off or passed to red permanently or abnormally, the traffic being disorganized as a result. Direct attacks to a large number of cars should also be envisaged, for instance simultaneously triggered by timed logical bombs, for instance forcing all vehicles to a halt at the same time. It is quite clear that a wave of such attacks would quickly result in a generalized loss of confidence towards the economical value of cars and might also indirectly harm the economy.

B.2.9 Mass terrorism

An augmentation of the attack motivation to harm the economic is mass terrorism. Most of the possible attacks to do this are already described before, but there are important differences:

- First, the scale of the attack is different because mass terrorism will probably target a large number of victims at a time.
- Another difference is that a terrorist organization will frequently accept to sacrifice some of its agents and even more frequently try to be identified while a classic attacker will do her very best to succeed without being caught and identified.
- In the same way the relation of expected results and involved resources, which are crucial for criminal attackers aiming at positive financial gain, the amount of allocated resources (financial but also in terms of man power, time, etc.) in case of mass terrorism is of minor importance.
- Finally, while an attack targeting single individuals is very unlikely, terrorists could try to cause huge traffic jams in order to harm a country's economy with the difference to that the effective harm is less important than the caused insecurity.

Due to all mentioned problems and differences and taking into account that the necessary spread of a system will need a long term, this attack should be examined in another study.

B.3 Possible attacks – Combining attack motivations and use cases

B.3.1 Force Green Wave/Getting traffic lights green ahead of the attacker

Based on the use cases about traffic information (from/to externals) one attack could consist in getting all traffic lights in front of the attacker to switch to green. Suppressing all halts will thereby increase the attacker's speed. There are several ways to do that: The attacker might have his car impersonate an emergency vehicle. Alternatively it is possible to directly tamper with the infrastructure in order to gain access to traffic signalling functions: this could be possible by exploiting poorly designed protocols. A physical attack to the infrastructure might also result in a similar result.

The attack tree based on this attack goal is shown in Figure 4 below.

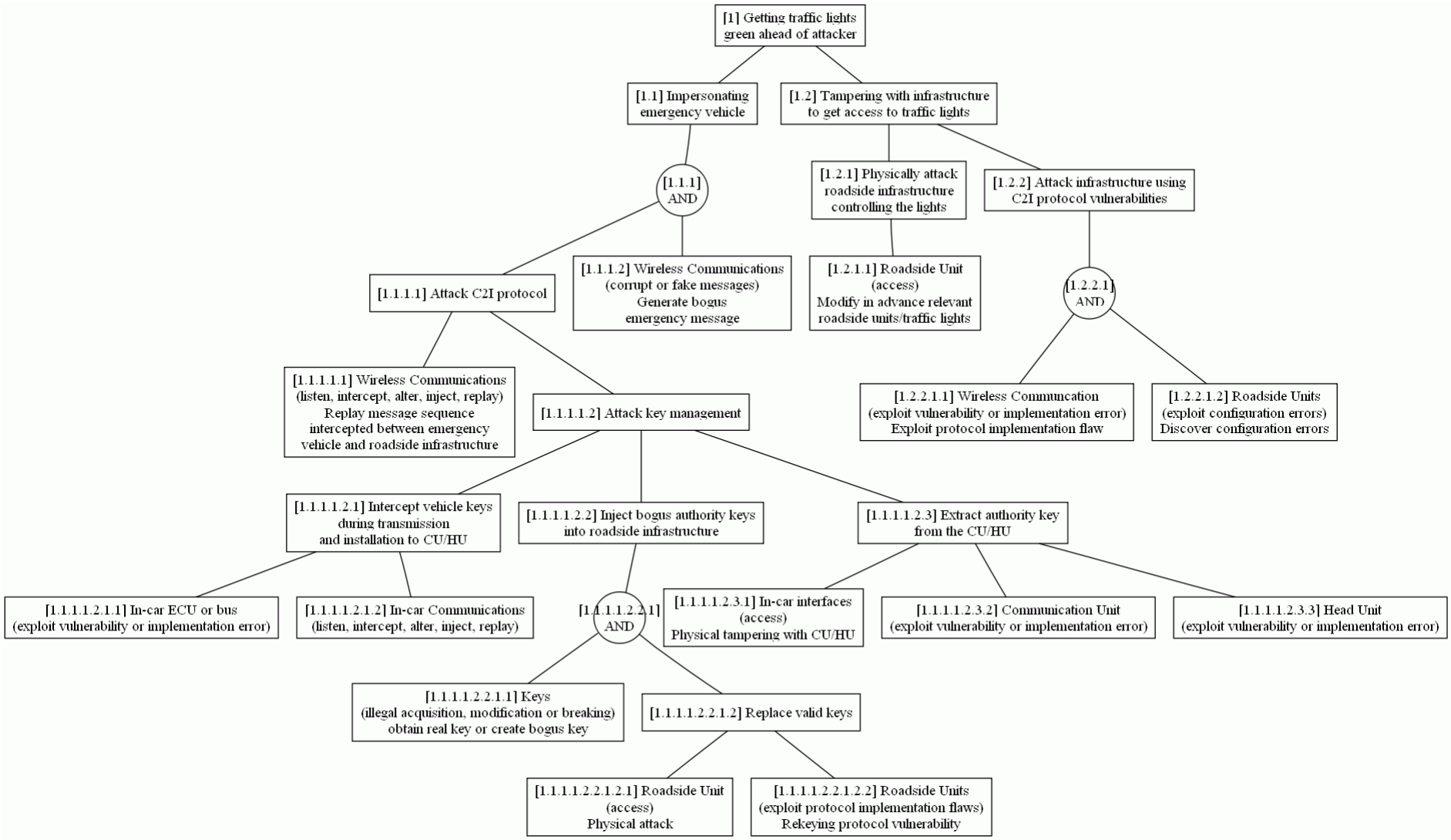


Figure 4 Attack tree 1: Force green lights ahead of attacker

B.3.2 Manipulate Speed Limits

The manipulation of speed limits is based on the use cases about traffic information or local danger warnings. In particular, it might be possible to tamper with the infrastructure so that other vehicles are notified of a lower speed limit. Speed limit may be changed arbitrarily to disorient drivers and to make them slow down, thereby slowing vehicles behind the attacker for instance. If speed limit enforcing equipment (e.g. radar) is accessible and may be tampered with, the attacker might set a higher speed limit. Also a physical attack to the infrastructure might be able to have a similar result.

The attack tree based on this attack goal is shown in Figure 5 below.

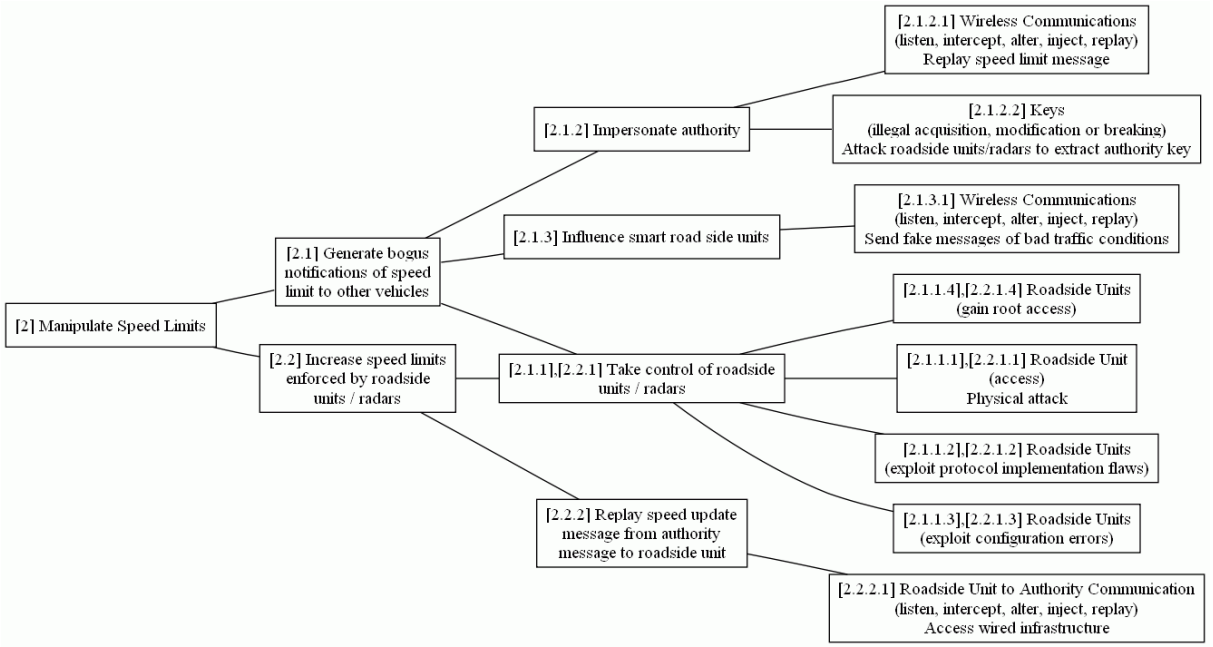


Figure 5 Attack tree 2: Manipulate speed limits

B.3.3 Manipulate Traffic Flow

To manipulate the traffic flow an attacker can also abuse traffic information or local danger warnings in different ways. One possibility is that the attacker might aim at re-direct other cars to alternative routes, thereby clearing any traffic jam in front of him (and likely creating more congestion elsewhere). This might again be made possible by impersonating an emergency vehicle, notably to send fake information about accidents in order to direct vehicles to alternative routes. The attacker might more importantly impersonate the infrastructure or tamper with it in order to send bogus information about the traffic ahead. Finally, the attacker might impersonate a chain of “fake cars“ and transmit their supposed position to the infrastructure and to nearby cars so that they all over-estimate the traffic at a given position on the road.

The attack tree based on this attack goal is shown in Figure 6.

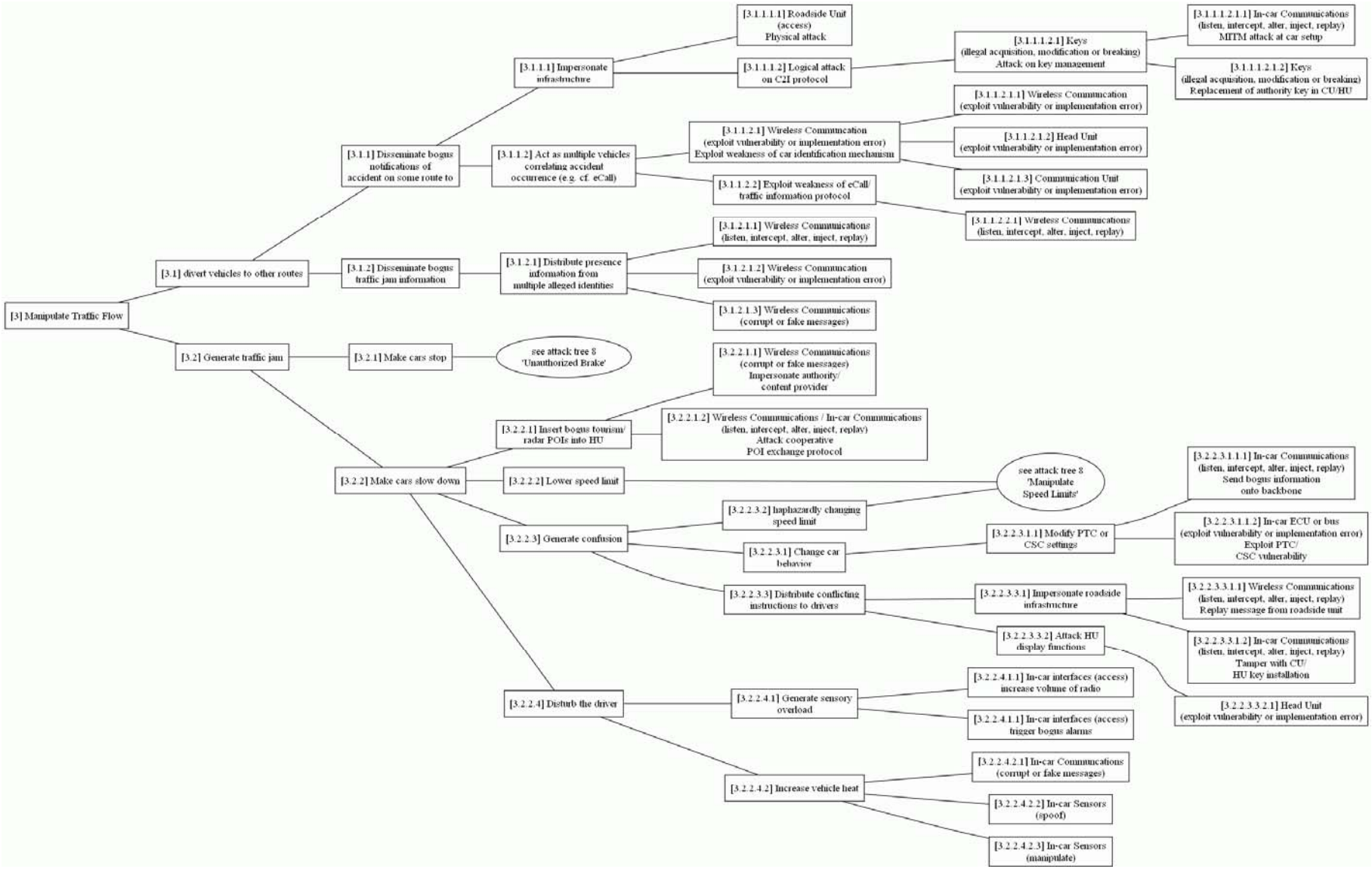


Figure 6 Attack tree 3: Manipulate traffic flow

B.3.4 Simulate Traffic Jam

The last case of misusing traffic information and local danger warning is to generate traffic congestion at a given place. This might be achieved by generating fake warning messages to make cars brake and slow down, or by tampering with the infrastructure and switching traffic lights randomly to red. Another possible attack is to misuse guided tours with point of interests in way, that someone create bogus points of interest (PoIs), either sightseeing/tourism-related, or traffic-related (like radars), to make cars slow down and to generate or increase congestion. This might be realised through impersonating the infrastructure or by attacking vehicular platforms directly.

The attack tree based on this attack goal is shown in Figure 7.

B.3.5 Tamper with Warning Message

On the one hand an attack to tamper with warning messages can be a consequence of an attack on the head unit. By gaining control over the HU an attacker could fake a warning message on the display and irritate or harm the driver by doing this. Another way to execute this attack is to delay or prevent warning messages that come from the local warning system (see for example “manipulate traffic flow”) or the brake info system (see “brake attacks“). A third way is an attack via the communication with the infrastructure or the infrastructure directly. One possibility is to spoof the GPS/Galileo signal send to the car. Another is to fake warning messages, which pretend to be sent by the infrastructure or to hack the infrastructure and use it for sending fake warnings. Finally one can attack the car physically and relay or fake message in the backbone.

The attack tree based on this attack goal is shown in Figure 8.

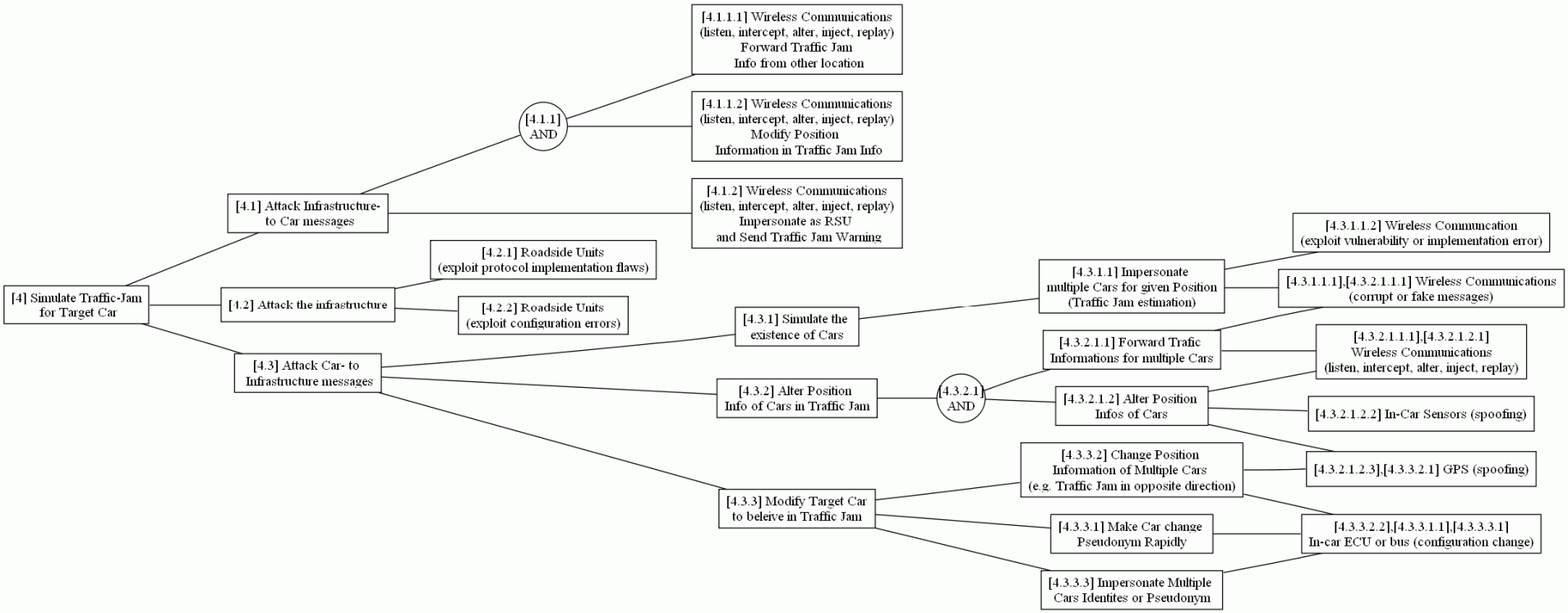


Figure 7 Attack tree 4: Simulate traffic jam

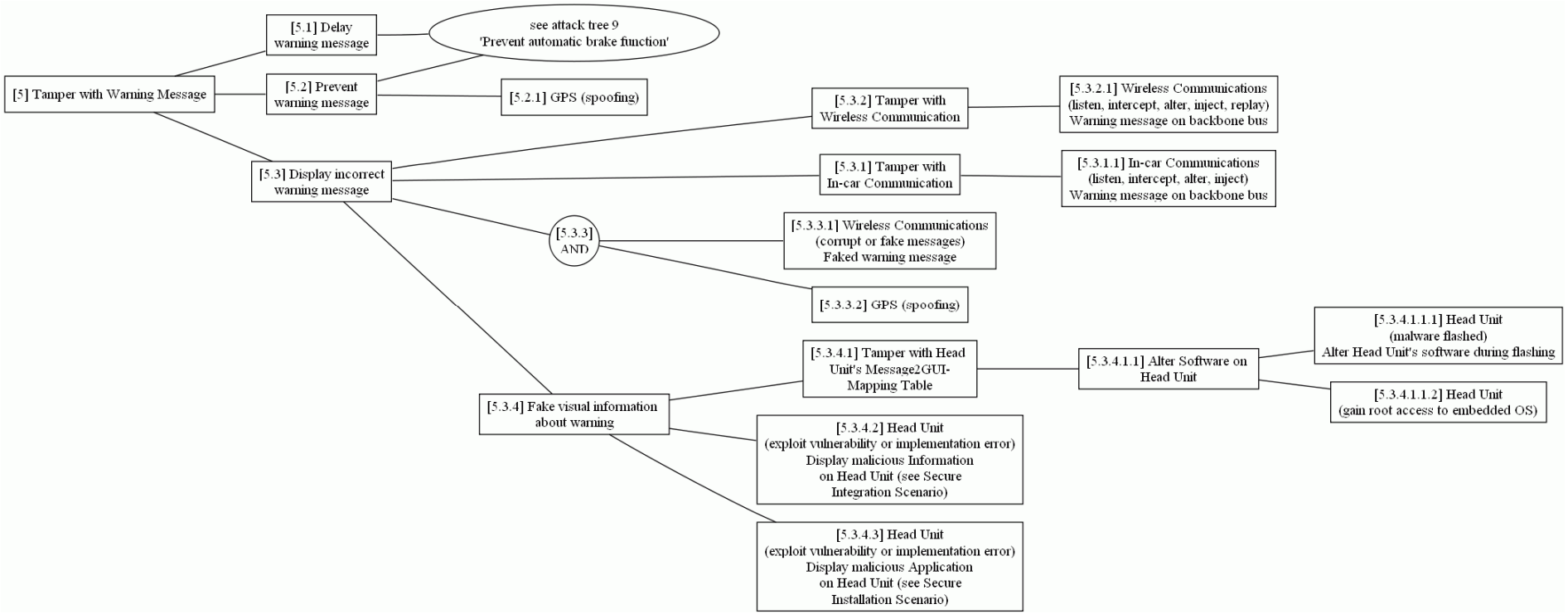


Figure 8 Attack tree 5: Tamper with warning messages

B.3.6 E-Call

Attacking the e-Call, which is intended to generate and send automatically the last positions of the vehicle (position chain) based on GPS / Galileo signals to the PSAP, can pursue different aims, either to trigger the e-Call without an accident, to degrade the quality of the service or to disable the service completely. Each of these attacks has different possible entry points. In the case of triggering an e-Call without accident the attacker can misuse the C2C brake info system to start an emergency braking and to start an e-Call. Another way to trigger the e-Call is to attack the function directly within the CU by sending corrupt data to initiate sending an e-Call. This can have a point of departure in all use cases involving the CU. To degrade the quality of the service an attacker can also misuse all use cases involving the CU. In this case the attack would aim to send imprecise or wrong information about the position. Another way to do this would be attacking the GPS module by jamming the signal outside of the car. If the attacker aims to interrupt the service completely all use cases that involve the CU can be abused. In this case the attacks would try to interrupt the communication to the service centre by jamming the signals or by a DoS attack on the CU. Another way would be to manipulate the CU in a way that it will try to contact a wrong number or non-existing service centre. A final way to interfere the service is an attack on the service centre, but this attack is not in the scope of EVITA.

The attack tree based on this attack goal is shown in Figure 9.

B.3.7 Engine DoS-Attack (Engine Refuse to Start)

One possibility to harm the driver or damage the reputation of a manufacturer can aim at preventing the car to start. To achieve this objective one could attack the Powertrain (PTC) and its devices. A possible attacker had to get access to these components of a car. The most likely use cases, which can involve this, are those on “remote diagnosis”, “remote flashing” or “flashing by OBD”. They allow corrupting the PTC or more directly the Engine controls (EC) by flashing firmware with corrupt code, changing important parameters or produce wrong communications. In all cases the PTC or EC would deny starting the car. Another way of attacking would be jamming the backbone.

The attack tree based on this attack goal is shown in Figure 10.

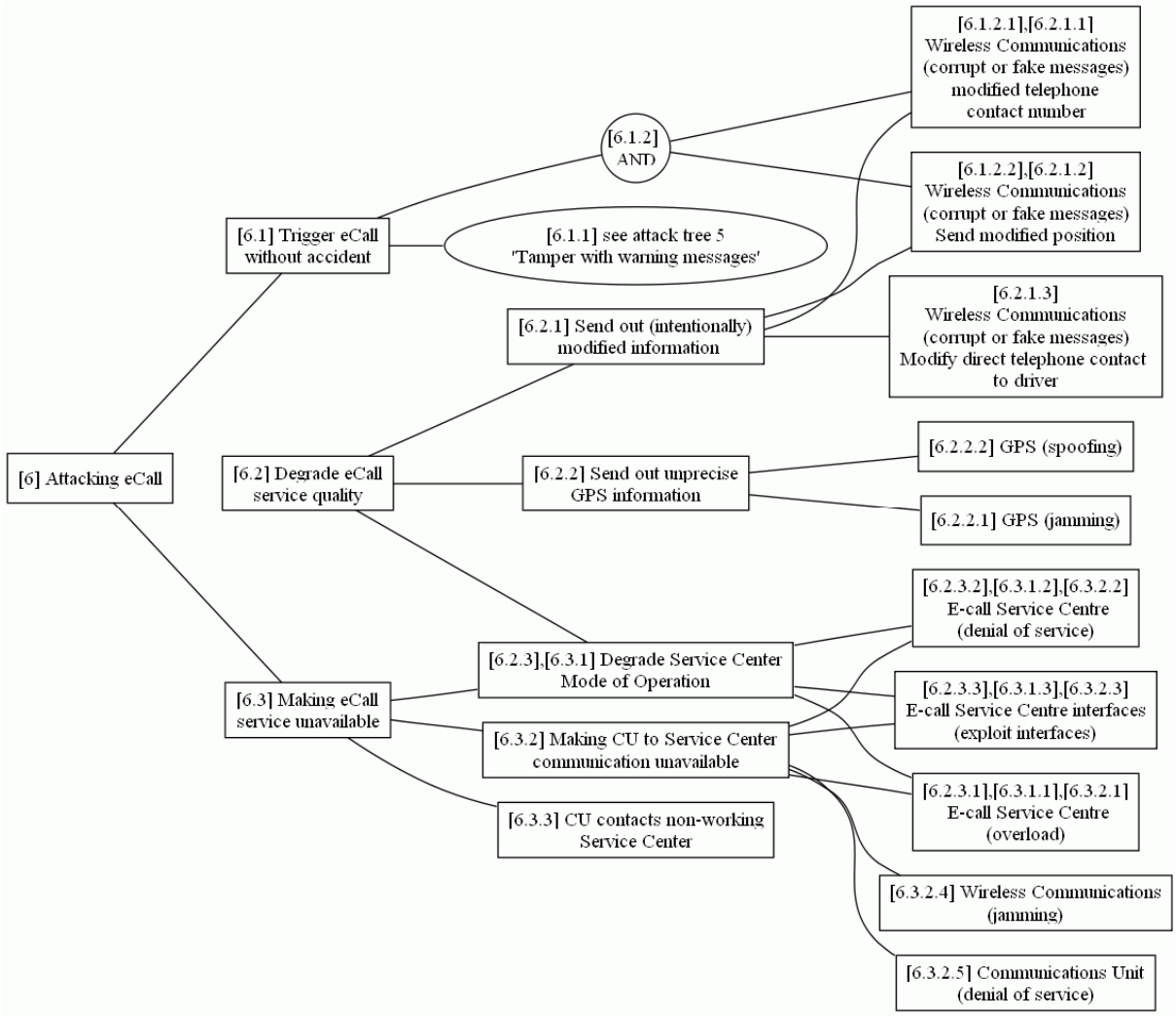


Figure 9 Attack tree 6: Attacking E-Call

B.3.8 Unauthorized Brake

Unauthorized braking can be the result of several possible attacks. One of these is an attack on the environment sensors, which can trigger a brake, as well as to manipulate the Chassis Safety Controller (CSC). While a direct attack of the sensors is only possible by flashing the firmware with malicious code (use cases “flashing” or “remote flashing”), the CSC can also be manipulated by exploiting implementation flaws or corrupt data. The most likely attack would involve faking a brake event in the direct environment of the car to produce this corrupt data. Therefore someone could misuse the C2C brake info to fake brake information from another car

The attack tree based on this attack goal is shown in Figure 11.

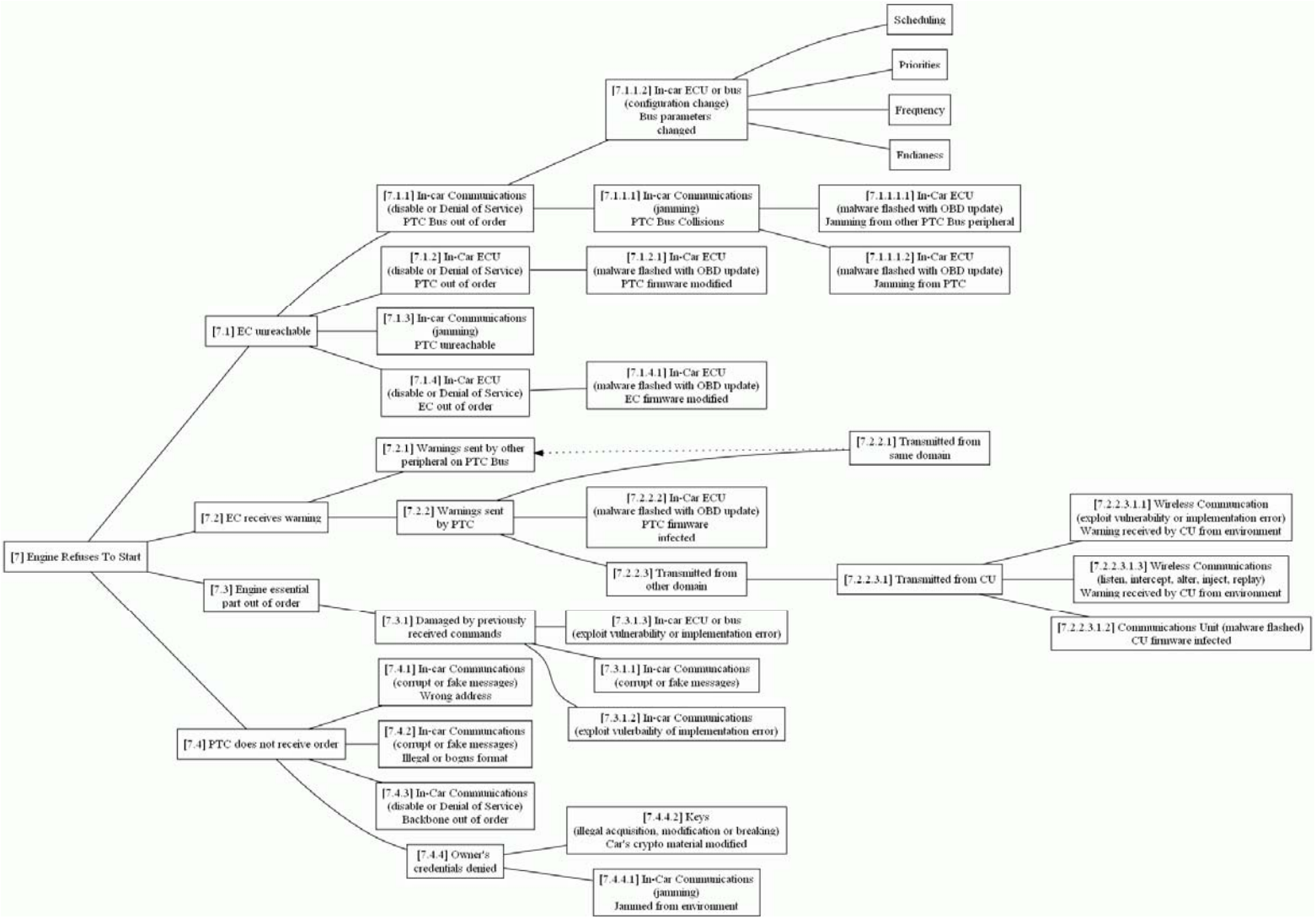


Figure 10 Attack tree 7: Engine refuses to start

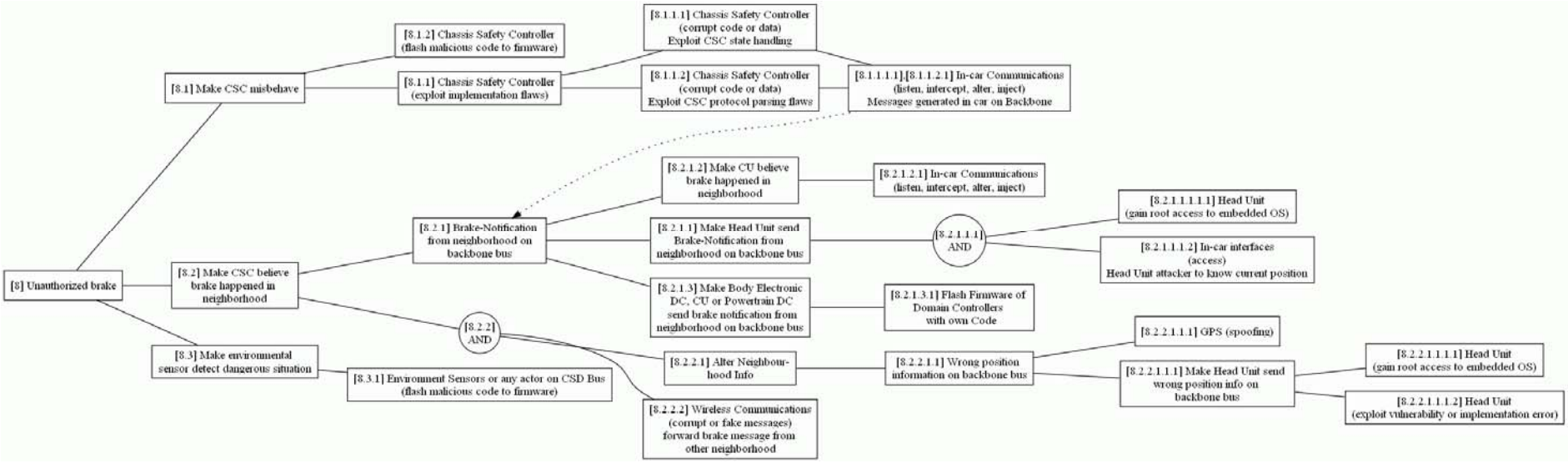


Figure 11 Attack tree 8: Unauthorized brake

B.3.9 Attacking Active Brake Function

Besides of initiating an unauthorized braking an offender could also try to attack an authorized braking event. This attack can be executed in different ways. For one thing the attacker can try to inhibit active braking completely; for another thing the attacker can try to delay the braking or at least to degrade the quality of the active brake. As in other attacks several use cases can constitute the starting point depending on the current circumstances, e.g. if the attacker has direct access to the car or the target of attack (inhibit, delay, degrade). One example, which is rather unlikely, is to corrupt or disable the environment sensors, sensors for ABS and ESP or the CSC (as described above). It seems easier (and more likely) to attack the CU or the CSC with a DoS attack to prevent or delay the computation/detection of events needed for the active braking system. This attack could abuse most of the use cases that involve the CU like the "integration of applications" or the "connection to external devices" such as mobile phones. Another possibility to reach this attack goal is jamming of the air interface, the CSC or Backbone bus.

The attack tree based on this attack goal is shown in Figure 12.

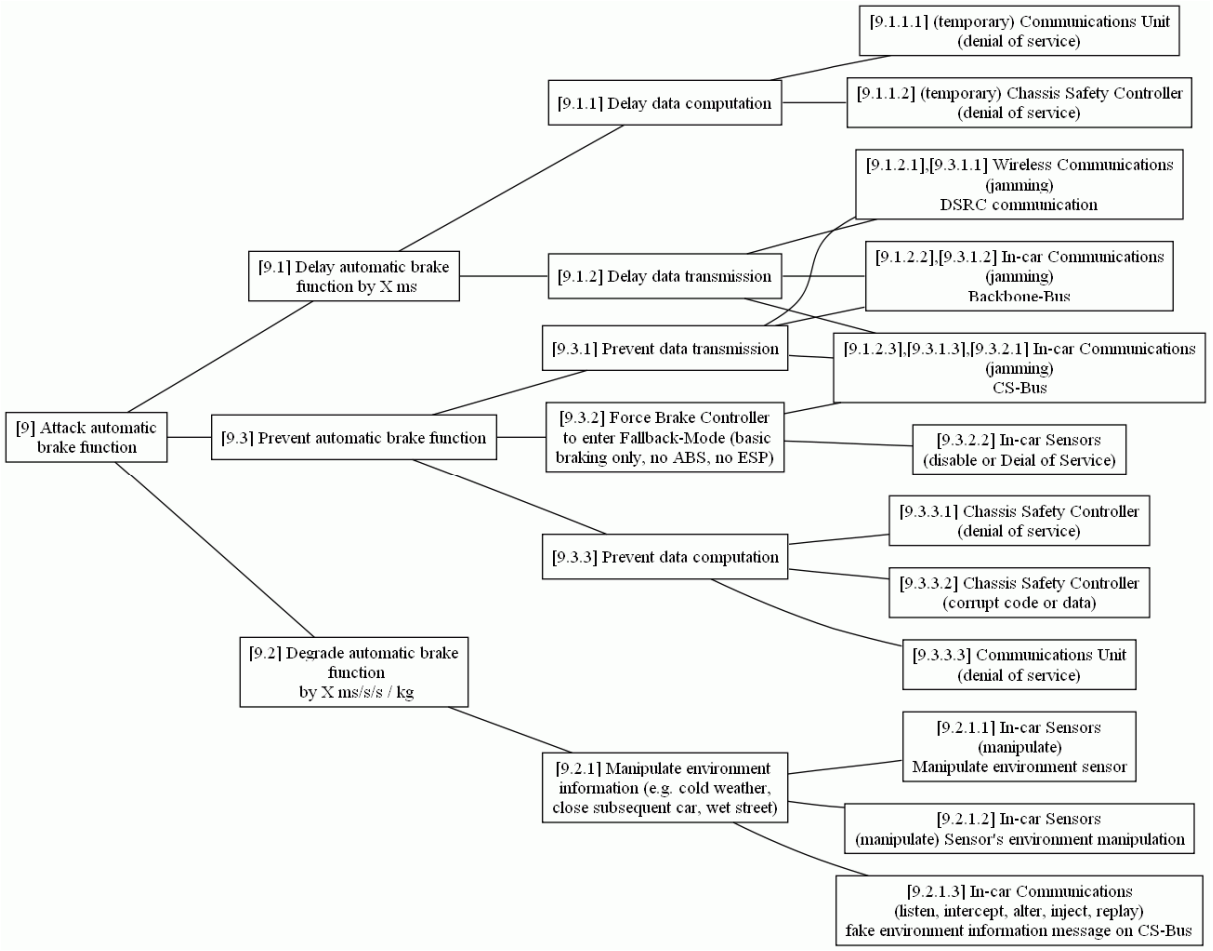


Figure 12 Attack tree 9: Attack active brake function

B.3.10 Attacking E-Toll

By attacking the e-Toll system one can pursue different objectives. First the attacker can try to harm another driver by denying the passage or increasing the toll, but it is also possible to misuse this function to receive a victim's personal data like bank account or credit card number. Another goal, maybe the most obvious, would be that attackers try to joy ride, i.e. to avoid toll payments for themselves. Attacking the CU and its components like the GPS system can carry out most of these attacks. For example the offender could manipulate the CU with corrupt keys to prevent the victim from passing a toll station. This manipulation can be done in several ways, for example by misusing a mobile device, which will be connected to the CU (Use Case "Personalise car") or by remote or hardwired flashing of the OBD. The same attack paths can be used to modify the GPS billing data with the aim to increase or decrease the toll payment. Another way to reduce payments is to jam or spoof the GPS signals send. Finally it might also be possible to misuse the OBD updates (use cases "Remote Flashing" or "OBD flashing)" or manipulate the CU to gain access to personal data.

Attacks on E-tolling are illustrated in Figure 13 to Figure 16.

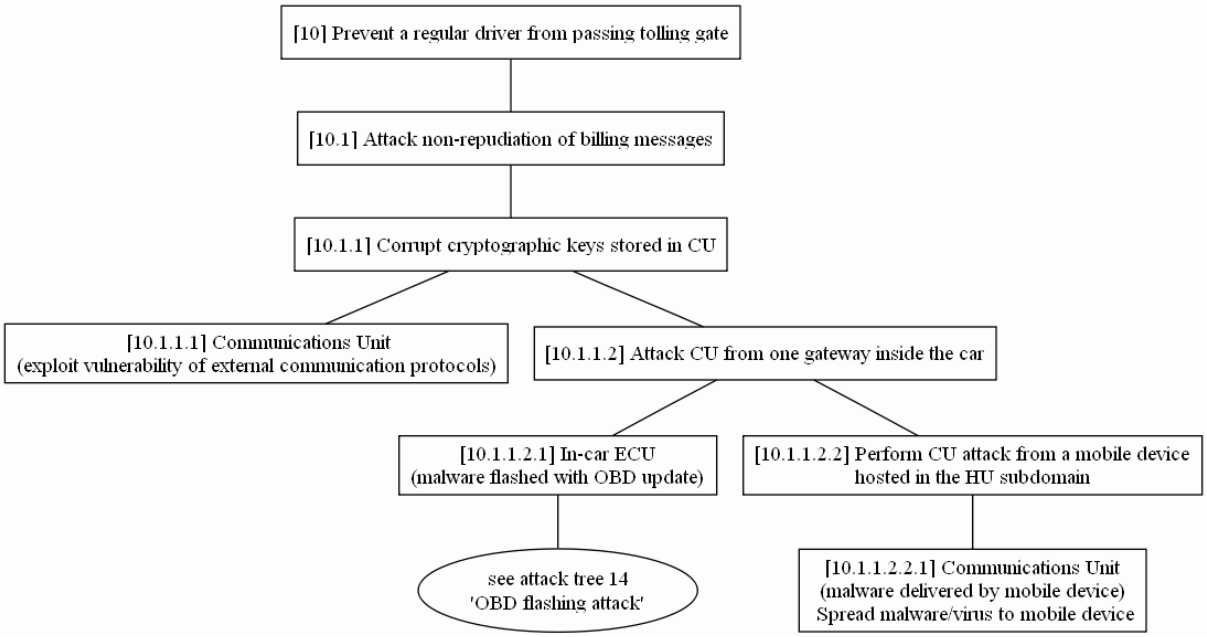


Figure 13 Attack tree 10: Prevent driver from passing toll gate

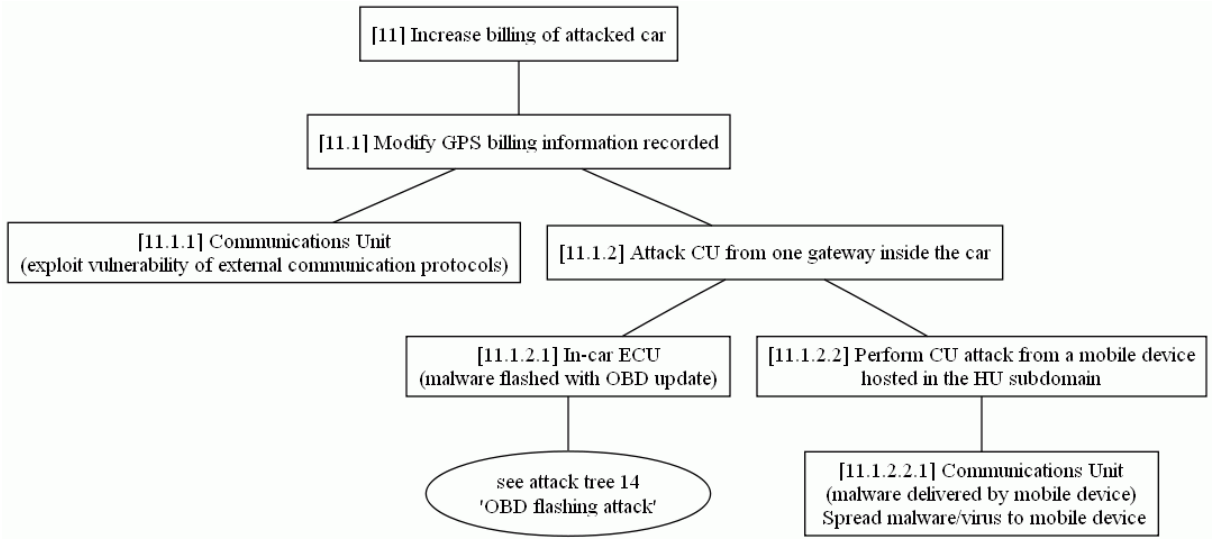


Figure 14 Attack tree 11: Increase driver's toll bill

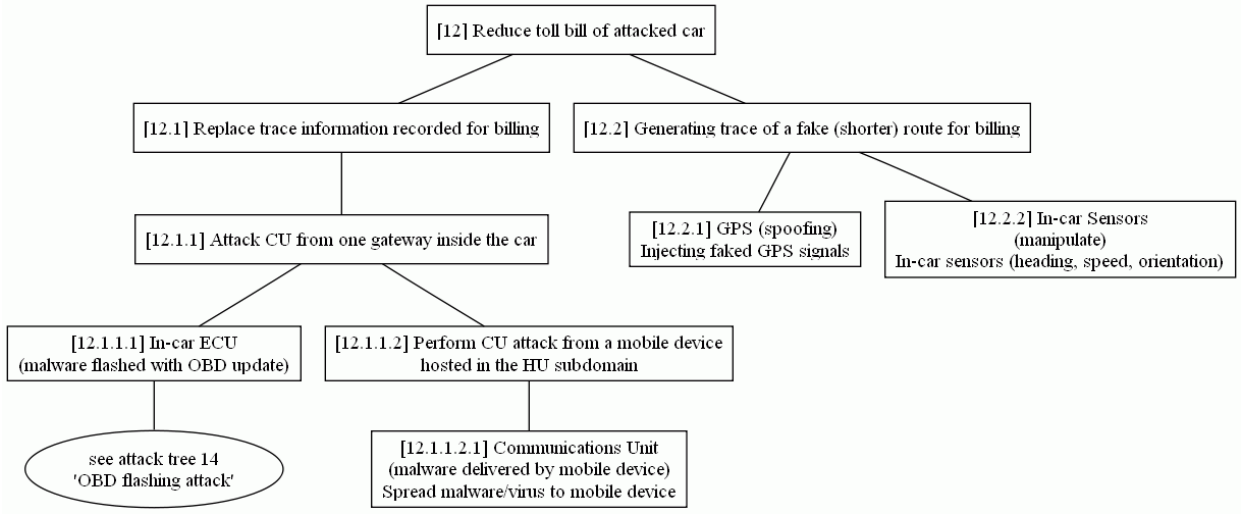


Figure 15 Attack tree 12: Reduce driver's toll bill

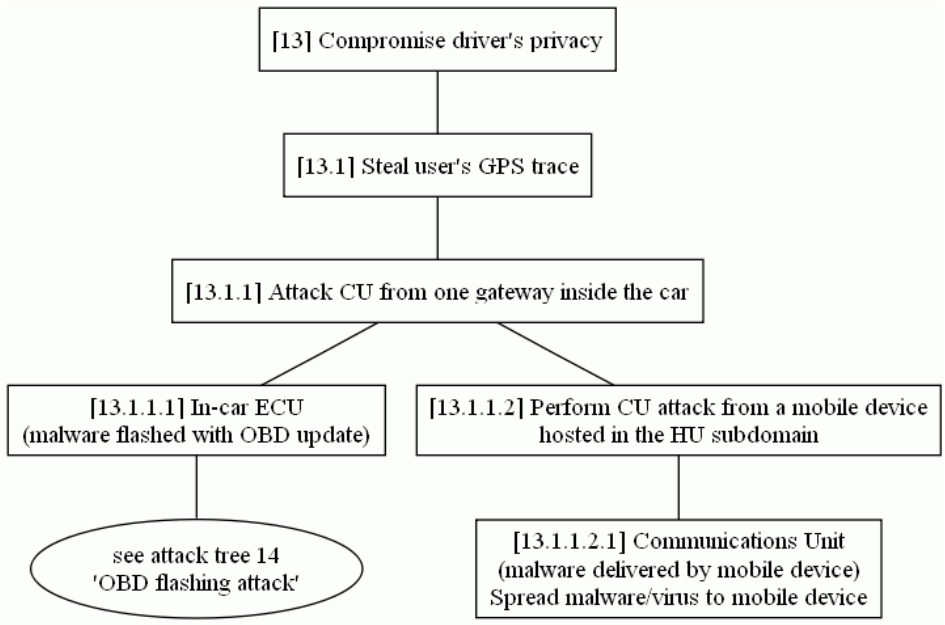


Figure 16 Attack tree 13: Compromise driver privacy

B.4 Attack Trees Detailing Asset Attacks

B.4.1 Flashing per OBD

There are two possible ways to attack the flashing via OBD. Most seriously an attacker would abuse the flashing itself in a workshop to install infected or modified firmware. If this is not possible the attacker can still try to gain access to the CU to interrupt or disturb the flashing process while the car is in the workshop. Another possible attack would try to gain access to the CU, for example within use cases like “Internet access” or “personalizing the car with an external device”, to exploit vulnerabilities in the communication protocols of diagnostic interface, CU and ECU to abort the next flashing per OBD.

The corresponding attack tree is shown in Figure 17.

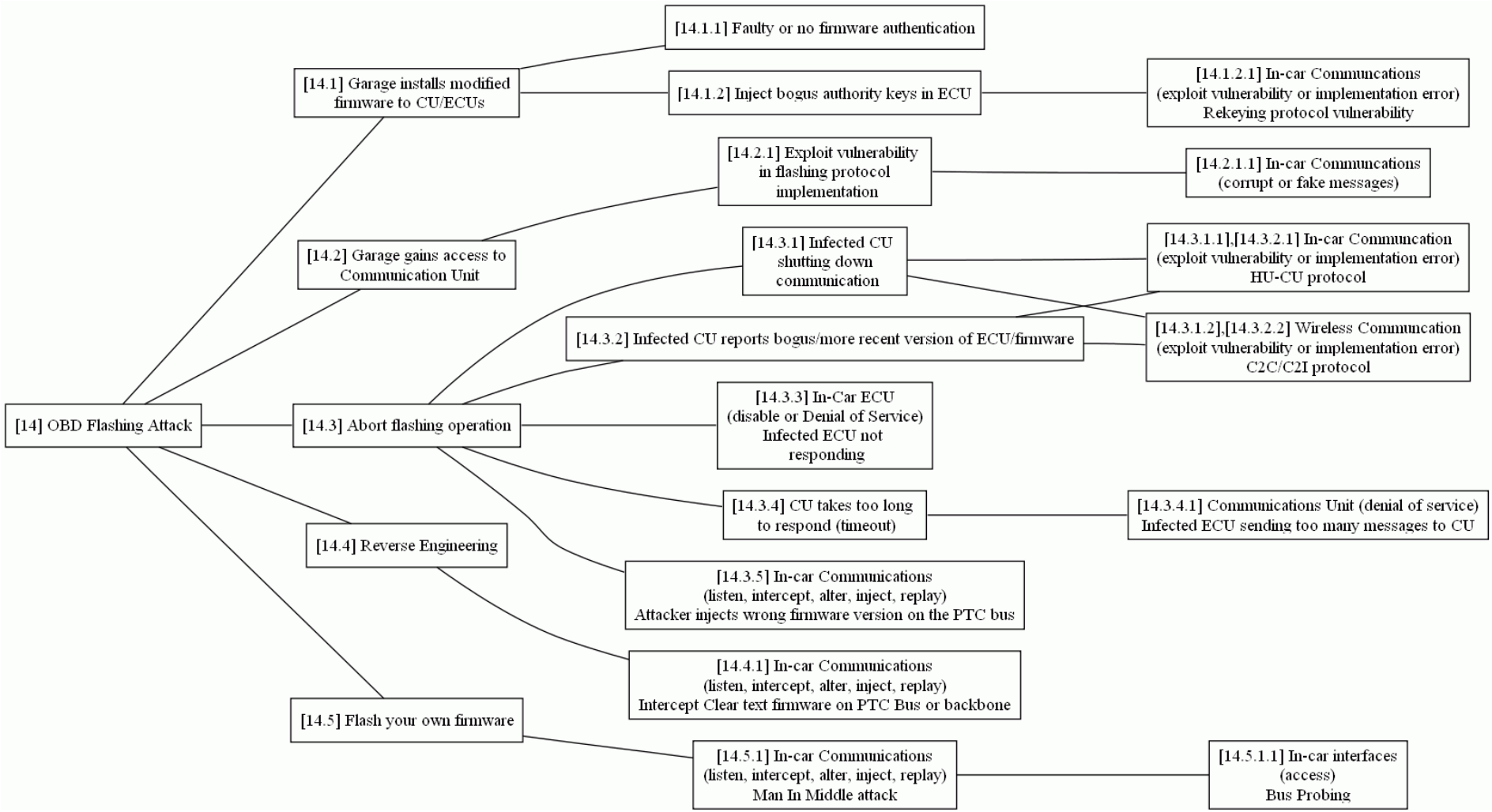


Figure 17 Attack tree 14: OBD flashing attack

B.4.2 Head Unit Attack

Several use cases include the possibility to carry out an attack on the head unit of the vehicle. First all use cases, which are directly related to the head unit, could be exploited. There are four use cases, which are connected either by Bluetooth (remote car control, personalize the car) or by USB (secure integration, installation of application). In these cases vulnerabilities of Bluetooth, USB or direct access can be gained and used for an attack on the head unit. Another way of attacking the head unit could be to start an attack via the CU, which is also part of several communications related use cases (such as remote flashing, remote diagnosis, traffic information and all use cases involving Internet connection). In all these cases access control is the key for the attack. It is defined as the ability to permit or deny the use of a particular resource. In general it is the basis for functionality that provides access to car internal and external resources or other entities by dint of units like registered mobile phones, trusted counterparts or the car itself. Examples for access control functionality within the use cases that use identification, authorization and access control based on managing of possession and location of registered mobile phones or trusted counterparts. Finally the head unit might be targeted by a physical attack. The corresponding attack tree is shown in Figure 18.

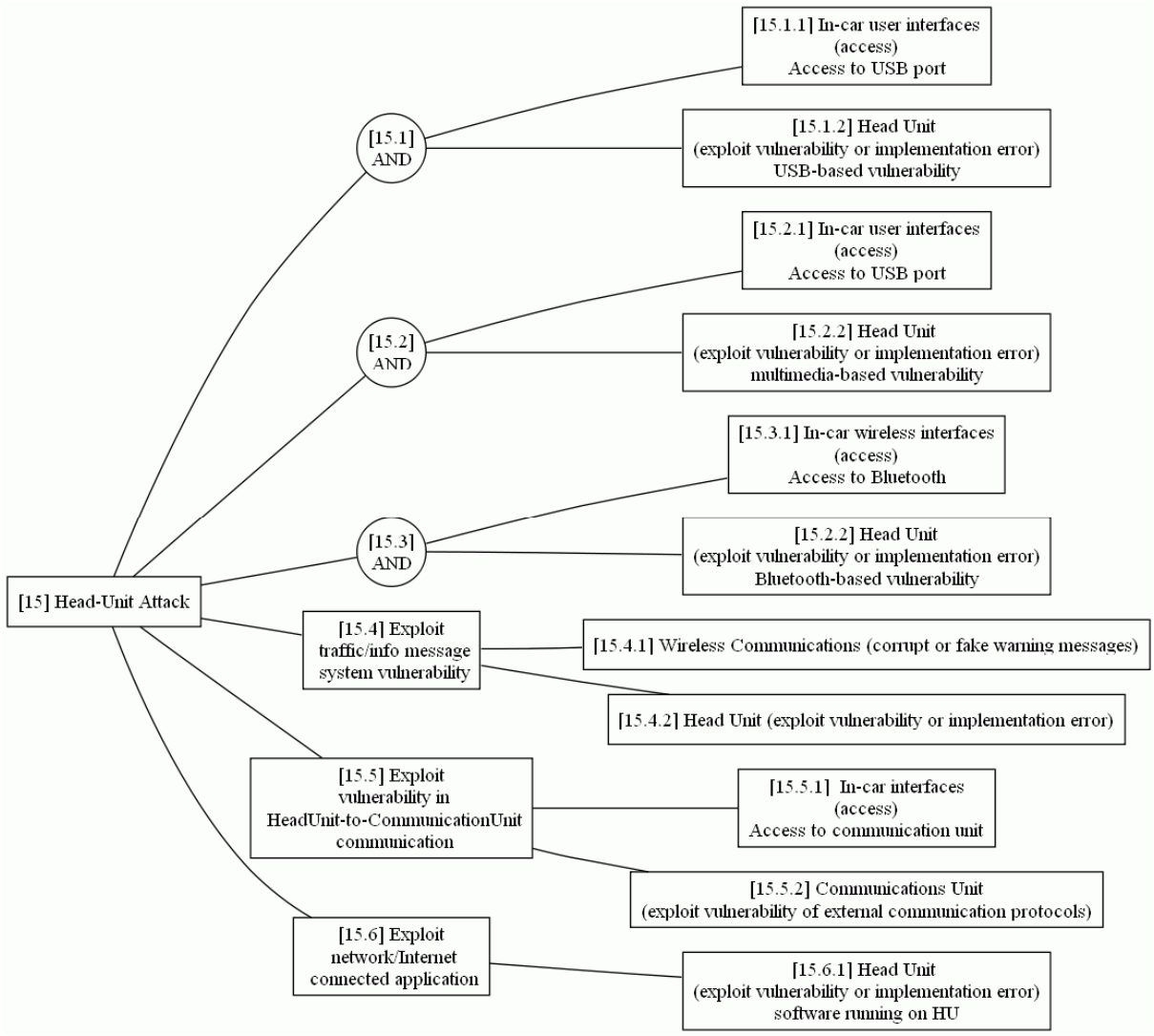


Figure 18 Attack tree 15: Head unit attack

Appendix C – Threat and risk analysis

C.1 Analysis methodology

C.1.1 Introduction

In order to identify the most relevant security requirement to be able to prevent or at least detect and contain a threat, we need to assess the level of "risk" posed by potential attacks. The risk of an attack is seen as a function of the possible severity (i.e. the cost and loss) of the attack for the stakeholders and the estimated probability of occurrence of a successful attack.

C.1.2 Notion of severity

In vehicle safety engineering processes the primary focus is on physical injuries that might be sustained by persons as the consequence of a safety hazard. Nonetheless, managing safety risks also has the secondary benefit of helping to protect the reputation (and hence the market share) of the vehicle manufacturer and their suppliers.

In considering security threats in the context of networked vehicles and ITS systems, physical safety is only one of a number of aspects that may be subject to "harm": other issues could include loss of privacy or unauthorized financial transactions threats. Furthermore, the impact of security threats may be more widespread than a single vehicle, and there is a wider range of stakeholders who may be influenced by the consequences of security hazards (i.e. other road users, ITS system operators, civil authorities, vehicle manufacturers and their suppliers). Therefore, a variety of factors need to be considered in the EVITA security and risk analysis.

In order to accommodate this more complex situation the classification proposed here (see Table 4) separates and categorizes different aspects of the consequences of possible security breaches. The starting point for this classification scheme is the safety severity classification of ISO/DIS 26262, which is based on the Abbreviated Injury Scale [19]. For the purposes of EVITA, this has been adapted and augmented to consider both the greater numbers of vehicles that may be involved and implications for aspects other than safety, including:

- **privacy** – identification and tracking of vehicles or individuals;
- **financial** – financial losses that may be experienced by individuals or ITS operators;
- **operational** – interference with vehicle systems and functions that do not impact on functional safety.

For example, it is possible that an attack has little or no impact on safety, but presents significant risks in terms of compromised driver privacy or loss of reputation for vehicle manufacturers. This approach results in a kind of "severity vector" with four components that may have different ratings. However, the components may translate to different relative risk levels, depending on the probability measures that are applied to assess the associated risk level.

Table 4 Proposed severity classification scheme for security threats

Security threat severity class	Aspects of security threats			
	Safety (S_S)	**Privacy (S_p)**	**Financial (S_F)**	**Operational (S_o)**
0	No injuries.	No unauthorized access to data.	No financial loss.	No impact on operational performance.
1	Light or moderate injuries.	Anonymous data only (no specific driver of vehicle data).	Low-level loss (~€10).	Impact not discernible to driver.
2	Severe injuries (survival probable). Light/moderate injuries for multiple vehicles.	Identification of vehicle or driver. Anonymous data for multiple vehicles.	Moderate loss (~€100). Low losses for multiple vehicles.	Driver aware of performance degradation. Indiscernible impacts for multiple vehicles.
3	Life threatening (survival uncertain) or fatal injuries. Severe injuries for multiple vehicles.	Driver or vehicle tracking. Identification of driver or vehicle, for multiple vehicles.	Heavy loss (~1000). Moderate losses for multiple vehicles.	Significant impact on performance. Noticeable impact for multiple vehicles.
4	Life threatening or fatal injuries for multiple vehicles.	Driver or vehicle tracking for multiple vehicles.	Heavy losses for multiple vehicles.	Significant impact for multiple vehicles.

C.1.3 Notion of probability of occurrence of successful attack (attack potential)

In IT security engineering, to be on the safe side, we must assume that each attack scenario that is possible and promises whatsoever small benefit will definitely be carried out by someone. The probability that an attack, once launched, will be successful depends on

- the "attack potential" of the attacker and
- the attack potential that the system under investigation is able to withstand (which the attack potential of the attacker needs to exceed).

If the attack potential of the attacker exceeds the attack potential that the system is able to withstand, then the system will definitely not withstand the attack and the attack will be successful.

The attack potential is well defined in [1][17]. The attack potential is a measure of the minimum effort to be expended in an attack to be successful. Essentially, the attack potential for an attack corresponds to the effort required creating and carrying out the attack. The higher the attackers' motivation, the higher efforts they may be willing to exert. There are multiple methods of representing and quantifying the influencing factors. The following factors should be considered during analysis of the attack potential [17]:

a) **Elapsed Time**: This is the total amount of time taken by an attacker to identify that a particular potential vulnerability may exist, to develop an attack method and to sustain effort required mounting the attack.

b) **Specialist Expertise**: This refers to the required level of general knowledge of the underlying principles, product types or attack methods.

c) **Knowledge of the system under investigation**: This refers to specific expertise in relation to the system under investigation. Though it is related to general expertise, it is distinct from that.

d) **Window of opportunity**: This has a relationship to the **Elapsed Time** factor. Identification and exploitation of a vulnerability may require considerable amounts of access to a system that may increase the likelihood of detection of the attack. Some attack methods may require considerable effort off-line, and only brief access to the target to exploit. Access may also need to be continuous or over a number of sessions.

e) **IT hardware/software or other equipment**: This refers to the equipment required to identify and exploit vulnerability.

In many cases these factors are not independent, but may be substituted for each other in varying degrees. For instance, expertise or equipment may be a substitute for time. Table 5 identifies the factors discussed above and, based on [17][18], associates numeric values with each level. Intermediate values to those in the table can also be chosen.

To determine for each path in an attack tree the attack potential required to identify and exploit it, sum up the appropriate values from Table 5 and apply Table 6 to classify the attack potential. Note that once an attack scenario has been identified and been exploited once, it may be exploited repeatedly with less effort than for the first time. Both phases, identification and exploitation, are considered in conjunction.

In this context the term “attack potential” is really describing the difficulty of mounting a successful attack, while for risk analysis purposes a probability measure is required. A high probability of successful attack is assumed to correspond to the “basic” attack potential, since many possible attackers will have the necessary attack potential. Conversely, a “high” attack potential suggests a lower probability of successful attacks, since the number of attackers with the necessary attack potential is expected to be comparatively small. Consequently, Table 6 also proposes an associated numerical scale that reflects the relative probability of success associated with the attack potential in a more intuitive manner. The “attack probability” measure (P) is higher for easier attacks that are associated with lower attack potentials, and lower for more difficult attacks associated with the higher attack potentials.

Table 5 Rating of aspects of attack potential

Factor	Level	Comment	Value
Elapsed Time	≤ 1 day		0
	≤ 1 week		1
	≤ 1 month		4
	≤ 3 months		10
	≤ 6 months		17
	> 6 months		19
	not practical	The attack path is not exploitable within a timescale that would be useful to an attacker.	∞
Expertise	Layman	Unknowledgeable compared to experts or proficient persons, with no particular expertise	0
	Proficient	Knowledgeable in being familiar with the security behaviour of the product or system type.	3[1]
	Expert	Familiar with the underlying algorithms, protocols, hardware, structures, security behaviour, principles and concepts of security employed, techniques and tools for the definition of new attacks, cryptography, classical attacks for the product type, attack methods, etc.	6
	Multiple experts	Different fields of expertise are required at an Expert level for distinct steps of an attack.	8
Knowledge of system	Public	e.g. as gained from the Internet	0
	Restricted	e.g. knowledge that is controlled within the developer organisation and shared with other organisations under a non-disclosure agreement	3
	Sensitive	e.g. knowledge that is shared between discreet teams within the developer organisation, access to which is constrained only to team members	7
	Critical	e.g. knowledge that is known by only a few individuals, access to which is very tightly controlled on a strict need-to-know basis and individual undertaking	11
Window of Opportunity	Unnecessary/ unlimited	The attack does not need any kind of opportunity to be realized because there is no risk of being detected during access to the target of the attack and it is no problem to access the required number of targets for the attack.	0
	Easy	Access is required for ≤ 1 day and number of targets required performing the attack ≤ 10.	1
	Moderate	Access is required for ≤ 1 month and number of targets required to perform the attack ≤ 100.	4
	Difficult	Access is required for > 1 month or number of targets required to perform the attack > 100.	10
	None	The opportunity window is not sufficient to perform the attack (the access to the target is too short to perform the attack, or a sufficient number of targets is not accessible to the attacker).	∞[2]
Equipment	Standard	readily available to the attacker	0
	Specialised	not readily available to the attacker, but acquirable without undue effort. This could include purchase of moderate amounts of equipment or development of more extensive attack scripts or programs.	4[3]
	Bespoke	not readily available to the public because equipment may need to be specially produced, is so specialised that its distribution is restricted, or is very expensive.	7
	Multiple bespoke	Different types of bespoke equipment are required for distinct steps of an attack.	9

[1] When several proficient persons are required to complete the attack path, the resulting level of expertise still remains "proficient".

[2] This indicates that the attack path is not exploitable due to other measures in the intended operational environment.

[3] If clearly different test benches consisting of specialised equipment are required for distinct steps of an attack, this should be rated as bespoke.

Table 6 Rating of attack potential and attack probability

Values	Attack potential required to identify and exploit attack scenario	Attack probability *P* (reflecting relative likelihood of attack)
0-9	Basic	5
10-13	Enhanced-Basic	4
14-19	Moderate	3
20-24	High	2
≥ 25	Beyond High	1

C.1.4 Estimating risk

For convenience in subsequent analysis, it is desirable to describe the attack trees in a tabular form that captures the points were severity, attack potential of individual steps, and combined attack potential for attack scenarios can be assigned, whilst compressing the level of detail to just that required to assess the combined attack potential (see Table 7). The aim of this is to achieve a more compact representation of the attack tree information by focussing on the "asset attacks", which can be assigned an attack potential in some way, and how they contribute to "attack objectives", where the severity of the attack consequences can be assessed. In this it is useful to retain the "attack methods", since these are described in terms of particular combinations of "asset attacks". However, both intermediate attack goals and the detailed breakdown associated with each of the asset attacks can be suppressed.

Table 7 Tabular representation of key elements of an attack tree

Attack Objective	Attack Method	Asset attack
A	A1	a & b
	A2	d
		e
		f
B	B1	a & b & c
		c & h
	B2	g

Tables representing the key elements of the attack trees can be augmented with the severity (**S**, a vector) for the attack objective and the estimated attack potential for the contributing asset attacks, using the numerical scale proposed in Table 6 to reflect the relative probability of a successful attack (*P*, a scalar). The relationships between the latter are then used to derive a combined attack potential for the particular attack method (*A*, a scalar).

If an attack method can be implemented using any one of a number of asset attacks (i.e. OR relationship) the combined attack probability (A_{OR}) is taken to be the **highest** of the attack probabilities (P_i) for the available asset attack options:

$$A_{OR}\left(P_i\right) = \max\left\{P_i\right\} \tag{C1}$$

(i.e. the attack probability of a set of alternative attack steps is as high as the highest of the possible alternatives).

Where the attack method requires a conjunction of asset attacks (i.e. AND relationship), the combined attack probability (A_{AND}) is taken to be the **lowest** of the attack probabilities (P_i) associated with the contributing asset attacks:

$$A_{AND}(P_i) = \min\{P_i\} \tag{C2}$$

(i.e. the attack probability of combined attack steps is only as high as the lowest of the required components).

When the attack method involves asset attacks combined using both AND and OR relationships the combined attack probability is built up using the rules (B1) and (B2) as appropriate. This process is illustrated in Table 8, for the example presented in Table 7.

Table 8 Attack tree of Table 7 augmented with risk analysis parameters

Attack Objective	Severity (S)	Attack Method	Risk level (R)	Combined attack potential (*A*)	Asset (attack)	Attack Probability (*P*)
A	$\mathbf{S_A}$	A1	$\mathbf{R_{A1}}(\mathbf{S_A},A_{A1})$	$A_{A1}=\min\{Pa,Pb\}$	a & b	Pa Pb
		A2	$\mathbf{R_{A2}}(\mathbf{S_A},A_{A2})$	$A_{A2}=\max\{Pd,Pe,Pf\}$	d	Pd
					e	Pe
					f	Pf
B	$\mathbf{S_B}$	B1	$\mathbf{R_{B1}}(\mathbf{S_B},A_{B1})$	$A_{B1}=\max[\min\{Pa,Pb,Pc\}, \min\{Pc,Ph\}]$	a & b & c	Pa Pb Pc
					c & h	Pc Ph
		B2	$\mathbf{R_{B2}}(\mathbf{S_B},A_{B2})$	$A_{B2}=Pg$	g	Pg

The risk level (**R**, a vector) is determined from the severity (**S**) associated with the attack objective and the combined attack probability (*A*) associated with a particular attack method. This is achieved by mapping the severity and attack probability to the risk using a "risk graph" approach. For severity aspects that are not safety related the risk graph maps two parameters (attack probability and severity) to a qualitative risk level. Combinations of severity and combined attack probability are mapped to a range of "security risk levels" (denoted *Ri*, where "*i*" is an integer) in Table 9 for non-safety security threats. The security risk level attributed to an attack increases with increasing severity and/or attack probability (the latter corresponding to lower attack potential).

Table 9 Proposed security risk graph for non-safety security threats (privacy, financial and operational)

Security Risk Level (R)		**Combined attack probability (A)**				
		A=1	A=2	A=3	A=4	A=5
Non-safety severity (S_i)	S_i=1	R0	R0	R1	R2	R3
	S_i=2	R0	R1	R2	R3	R4
	S_i=3	R1	R2	R3	R4	R5
	S_i=4	R2	R3	R4	R5	R6

Where the severity vector includes a non-zero safety component, the risk assessment may include an additional probability parameter that represents the potential for the driver to influence the severity of the outcome. In the MISRA Safety Analysis Guidelines [3] and ISO/DIS 26262 [2] this possibility is reflected in a qualitative measure referred to as "controllability" (see Table 10).

Table 10 Classification for controllability of safety hazards

Class	Meaning
C1	Despite operational limitations, avoidance of an accident is normally possible with a normal human response.
C2	Avoidance of an accident is difficult, but usually possible with a sensible human response.
C3	Avoidance of an accident is very difficult, but under favourable circumstances some control can be maintained with an experienced human response.
C4	Situation cannot be influenced by a human response.

In order to include the additional parameter (controllability) in the assessment of safety-related security risks it is necessary to use of a different risk graph as proposed in Table 11, which maps three parameters (severity, attack probability, and controllability) to qualitative risk levels. The class "R7+" that is used in Table 11 denotes levels of risk that are unlikely to be considered acceptable, such as safety hazards with the highest severity classes and threat levels, coupled with very low levels of controllability.

Table 11 Proposed security risk graph for safety-related security threats

Controllability (C)	Safety-related Severity (S_S)	Combined Attack Probability (A)				
		A=1	A=2	A=3	A=4	A=5
C=1	S_S=1	R0	R0	R1	R2	R3
	S_S=2	R0	R1	R2	R3	R4
	S_S=3	R1	R2	R3	R4	R5
	S_S=4	R2	R3	R4	R5	R6
C=2	S_S=1	R0	R1	R2	R3	R4
	S_S=2	R1	R2	R3	R4	R5
	S_S=3	R2	R3	R4	R5	R6
	S_S=4	R3	R4	R5	R6	R7
C=3	S_S=1	R1	R2	R3	R4	R5
	S_S=2	R2	R3	R4	R5	R6
	S_S=3	R3	R4	R5	R6	R7
	S_S=4	R4	R5	R6	R7	R7+
C=4	S_S=1	R2	R3	R4	R5	R6
	S_S=2	R3	R4	R5	R6	R7
	S_S=3	R4	R5	R6	R7	R7+
	S_S=4	R5	R6	R7	R7+	R7+

C.1.5 Requirements for countermeasures

The risk analysis based on the attack trees provides the rationale for developing security requirements.

In safety engineering, the opportunities for mitigating the severity of the outcome of a hazardous situation (e.g. the use of airbags to reduce injury levels) are often limited. The most common approach for risk reduction is therefore to attempt to reduce "exposure" to the hazard. This is also likely to be true of security threats.

The attack trees provide a convenient basis for the systematic evaluation of possible attack methods and "cutting branches" from the tree is a possible mechanism for identifying requirements for specific countermeasures. The selection of branches to be cut can be prioritised based on risk levels and attack potential:

- where a number of possible attack objectives may achieve the attack goal, the attack objective with the highest perceived risk level is the priority for countermeasures to reduce the risk level for the attack objective;
- where a number of possible attack methods may lead to the same attack objective, the attack method with the highest perceived attack probability (i.e. lowest attack potential) is the priority for countermeasures to reduce the risk level for the attack objective;
- where a number of asset attacks may lead to the same attack objective, the asset attack with the highest perceived attack probability (i.e. lowest attack potential) is the priority for countermeasures to reduce the risk level for the attack objective.

Eliminating the asset attacks judged to have the highest attack probability (i.e. lowest attack potential) reduces the threat level for the associated attack method, and if the attack probability for this attack method dominates the risk level for the associated attack objective then the attack objective risk level will also be reduced.

Since the functions investigated all assume a common basic architecture, it is likely that common patterns will arise in the attack trees derived from the dark-side scenarios analysis. Consequently, the repeated occurrence of particular attack patterns in attack trees is a further indicator for prioritising countermeasures that are likely to provide favourable cost-benefit properties. However, the expected cost of the proposed countermeasures also needs to be taken into account in selecting specific security requirements.

C.2 EVITA Risk Analysis

C.2.1 Attack potential

Table 12 summarizes estimates for the "attack potential", together with the underlying estimates for the influencing factors, for various attacks identified from the attack trees. The estimates are based on as-is automotive on-board networks, prior to the introduction of security measures.

Table 12 Evaluation of required attack potential for asset attacks identified from attack trees

Attack tree node number	Asset (attack)	Elapsed time	Expertise	Knowledge of system	Window of opportunity	Equipment	Required attack potential	
							Value	Rating
[3.2.2.4.2.3], [9.3.2.2], [9.2.1.1], [9.2.1.2], [12.2.2]	In-car Sensors (external manipulation of sensor input)	0	0	3	0	0	3	Basic
[6.2.2.1],	GPS (jamming)	0	0	0	0	4	4	Basic
[9.1.2.1], [9.3.1.1]	Wireless Communications (jamming)	1	3	0	0	4	8	Basic
[1.1.1.2], [3.1.2.1.3], [3.2.2.1.1], [4.3.1.1.1], [4.3.2.1.1.1], [5.3.3.1], [8.2.2.2], [15.4.1]	Wireless Communications (corrupt or fake messages and information)	1	3	0	0	4	8	Basic
[9.3.2.2]	In-car Sensors (disable or Denial of Service)	4	0	0	1	4	9	Basic
[4.3.2.1.2.3], [4.3.3.2.1], [6.2.2.2], [5.2.1], [5.3.3.2], [8.2.2.1.1.1], [12.2.1]	GPS (spoofing)	4	3	0	0	4	11	Enhanced-Basic
[6.3.2.2], [9.1.1.1], [9.3.3.3],	Communications Unit (denial of service)	0	3	3	1	4	11	Enhanced-Basic
[7.4.3]	In-car Communications (disable or Denial of Service)	4	3	3	1	0	11	Enhanced-Basic
[15.3.1]	In-car wireless interfaces for short range communication (access)	4	0	3	1	4	12	Enhanced-Basic
[9.1.2.2], [9.3.1.2], [9.1.2.3], [9.3.1.3], [9.3.2.1], [7.1.3], [7.4.4.1]	In-car Communications (jamming)	4	3	0	1	4	12	Enhanced-Basic
[1.1.1.1.2.2.1], [3.2.2.4.1.1], [3.2.2.4.1.2], [8.2.1.1.1.2], [15.5.1]	In-car interfaces (access – exploit vulnerabilities, introduce bogus data)	0	6	3	1	4	14	Moderate
[15.1.1], [15.2.1]	In-car user hardware interfaces (access)	4	3	3	1	4	15	Moderate
[1.1.1.1.2.2.1.2.1] [1.2.1.1], [2.1.1.1], [3.1.1.1.1]	Roadside Unit (access)	4	6	3	1	4	18	Moderate

Attack tree node number	Asset (attack)	Elapsed time	Expertise	Knowledge of system	Window of opportunity	Equipment	Required attack potential	
							Value	Rating
[6.2.3.3], [6.3.1.3], [6.3.2.3]	E-call Service Centre interfaces (exploit interfaces)	4	6	3	1	4	18	Moderate
[1.1.1.1.2.2.2]	In-car Communications (exploit vulnerability or implementation error)	4	3	3	4	4	18	Moderate
[1.1.1.1.2.3.1]	In-car interfaces (access – physical tampering)	4	6	3	1	7	21	High
[5.3.4.1.1.2], [8.2.1.1.1.1.1], [8.2.2.1.1.1.1]	Head Unit (gain root access to embedded OS)	10	3	0	4	4	21	High
[6.2.3.1], [6.3.1.1], [6.3.2.1]	E-call Service Centre (overload)	4	0	3	10	4	21	High
[9.1.1.2], [9.3.3.1],	Chassis Safety Controller (denial of service)	10	3	3	1	4	21	High
[7.1.2]	In-car ECU (disable or Denial of Service)	10	3	3	1	4	21	High
[4.3.3.2.2], [4.3.3.1.1], [4.3.3.3.1], [7.1.1.2],	In-car ECU or bus (configuration change)	10	3	3	1	4	21	High
[10.1.1.2.2], [11.1.1.2.2], [12.1.1.2], [13.1.1.2.1]	Communications Unit (malware delivered by mobile device)	10	6	3	4	0	23	High
[3.1.1.2.1.3], [7.2.2.3.1.1], [10.1.1.1], [11.1.1.1], [15.5.2]	Communications Unit (exploit vulnerability of external communication protocols)	10	6	3	0	4	23	High
[1.1.1.1], [2.1.2.1], [2.1.3.1], [3.1.1.2.2.1], [3.1.2.1.1], [3.2.2.1.2], [3.2.2.3.3.1.1], [4.1.1.1], [4.1.1.2], [4.1.2], [4.3.2.1.1.1], [4.3.2.1.2.1]	Wireless Communications (listen, intercept, alter, inject, replay)	10	6	3	0	4	23	High

Attack tree node number	Asset (attack)	Elapsed time	Expertise	Knowledge of system	Window of opportunity	Equipment	Required attack potential	
							Value	Rating
[1.1.1.1], [2.1.2.1], [2.1.3.1], [3.1.1.2.1.1], [3.1.2.1.2], [4.3.1.1.2], [6.1.2.1], [6.2.1.1], [6.1.2.2], [6.2.1.2], [6.2.1.3], [6.1.2.3], [7.2.2.3.1.1]	Wireless Communication (exploit vulnerability or implementation error)	10	6	3	0	4	23	High
[3.2.2.4.2.1], [7.3.1.1]. [7.4.1]	In-car Communications (corrupt or fake messages)	10	6	3	0	4	23	High
[2.2.2.1]	Roadside Unit to Authority Communication (listen, intercept, alter, inject, replay)	10	6	3	0	4	23	High
[3.2.2.4.2.2], [4.3.2.1.2.2]	In-car Sensors (spoof)	10	3	3	4	4	24	High
[1.2.2.2], [2.1.1.3], [4.2.2]	Roadside Units (exploit configuration errors)	10	6	3	0	7	26	Beyond High
[6.2.3.2], [6.3.1.2], [6.3.2.2]	E-call Service Centre interfaces (denial of service)	4	6	3	10	4	27	Beyond High
[2.1.1.4], [2.2.1.4]	Roadside Units (gain root access)	10	6	3	4	4	27	Beyond High
[1.1.1.1.2.1.1], [7.3.1.3]	In-car ECU or bus (exploit vulnerability or implementation error)	10	3	3	4	7	27	Beyond High
[8.1.1.1], [8.1.1.2], [9.3.3.2],	Chassis Safety Controller (corrupt code or data)	10	6	7	1	4	28	Beyond High
[7.2.1]	Powertrain Peripherals (corrupt code or data)	10	6	7	1	4	28	Beyond High
[2.1.1.2], [4.2.1],	Roadside Units (exploit protocol implementation flaws)	10	6	7	0	7	30	Beyond High

Attack tree node number	Asset (attack)	Elapsed time	Expertise	Knowledge of system	Window of opportunity	Equipment	Required attack potential	
							Value	Rating
[1.1.1.1.2.3.2], [3.1.1.2.1.2], [3.2.2.3.3.2.1], [5.3.4.2], [5.3.4.3], [8.2.2.1.1.1.2], [15.1.2], [15.2.2], [15.3.2], [15.4.2], [15.6.1]	Head Unit (exploit vulnerability or implementation error)	10	6	6	4	7	33	Beyond High
[3.1.1.1.2.1.2], [7.4.4.2], [2.1.2.2]	Keys (illegal acquisition, modification or breaking)	17	6	7	4	1	35	Beyond High
[3.1.1.1.2.1.1], [3.2.2.1.3], [3.2.2.3.1.1.1], [3.2.2.3.3.1.2], [5.3.2.1], [5.3.1.1], [4.1.1.1], [4.1.1.2], [4.1.2], [8.1.1.1.1], [8.1.1.2.1], [8.2.1.2.1], [9.1.2], [9.3.1], [9.2.3.1]	In-car Communications (listen, intercept, alter, inject, replay)	17	6	6	4	4	37	Beyond High
[8.3.1]	Environment Sensors (flash malicious code to firmware)	17	6	7	4	7	41	Beyond High
[7.2.2.3.1.2], [10.1.1.2.1], [11.1.1.2.1], [12.1.1.1], [13.1.1.1.1]	Communications Unit (malware flashed with OBD update)	17	6	7	4	7	41	Beyond High
[8.1.2],	Chassis Safety Controller (flash malicious code to firmware)	17	6	7	4	7	41	Beyond High
[5.3.4.1.1.1]	Head Unit (malware flashed)	17	6	7	4	7	41	Beyond High
[7.1.2.1], [7.1.4.1], [7.2.2.2]	In-Car ECU (malware flashed with OBD update)	17	6	7	4	7	41	Beyond High
[7.2.2.2]	Powertrain controller (malware flashed with OBD update)	17	6	7	4	7	41	Beyond High

Note that Table 12 includes two attack methods on wireless and in-car communications (i.e. “corrupt or fake messages” and “listen, intercept, alter, inject, replay”) that have similar descriptions but are allocated very different attack potentials. The reasoning behind these differences is that while the attacker does not need to be in the communication path in order to send corrupt or fake messages, in order to intercept or inject packets the attacker must be in

the communication path and able to “hide” the genuine packets from the legitimate receiver. The later case is more difficult and requires a higher level of knowledge and preparation. Furthermore, the attacker does not need to be aware of the correct usage of the communication protocol in order to send corrupt or fake messages, but to inject or replay packets (that have to be accepted by the receiver) the attacker has to use a potentially proprietary communications protocol correctly (this covers all phases of communication: connecting, disconnecting, sending and requesting data). The later case is again more difficult and requires a higher level of knowledge and preparation from the attacker.

C.2.2 Attack active brake function

This attack is derived from the use case “Safety Reaction: Active Brake” [5]. The risk analysis table (see Table 13) is based on the corresponding attack tree (Figure 12).

Where an attack method could be implemented by a number of alternative means, the asset attack probabilities shown in bold are the most significant component for the resulting combined attack probability. In cases involving the AND relationship, the least significant asset attack probabilities for the combination are shown in italics. Consequently, the typeface and value assigned to the asset attack probabilities in the risk analysis tables give a visual indication of the priorities for risk reduction measures.

Loss of the active braking function is not expected to result in an additional safety hazard, since it is assumed that drivers will be able to respond to driving hazards by conventional manual braking. However, widespread loss of functionality may be detrimental to the reputation of this function, and thereby to vehicle manufacturers and system suppliers, suggesting an operational severity rating S_O=2.

In this example, an attack method involving an asset attack rated with attack probability P5 is identified for each attack objective (see Table 13). Since the asset attacks associated with each of the attack methods are all simple alternatives (i.e. OR relationship), the worst-case threat level for all of the attack objectives is simply the highest attack probability (*P5*). This translates to a risk level R4 since the severity S_O=2 is not safety related (see Table 9), indicating a moderate commercial risk.

Table 13 Risk analysis for "Attack Active Brake Function"

Attack Objective	Severity (S)	Attack Method	Risk level (R)	Combined attack probability (*A*)	Asset (attack)	Attack probability (*P*)
9.1 Delay active braking (e.g. by *x* ms)	S_S=0 S_P=0 S_F=0 S_O=2	Delay computation	R_O=R3	4	9.1.1.2 Chassis Safety Controller (denial of service)	2
					9.1.1.1 Communications Unit (denial of service)	**4**
		Delay data transmission	R_O=R4	5	9.1.2.1 Wireless Communications (jamming)	**5**
					9.1.2.2 Backbone Bus (jamming)	4
					9.1.2.3 Chassis Safety Bus (jamming)	4
9.3 Prevent active braking	S_S=0 S_P=0 S_F=0 S_O=2	Prevent computation	R_O=R3	4	9.3.3.1 Chassis Safety Controller (denial of service)	2
					9.3.3.2 Chassis Safety Controller (corrupt code or data)	1
					9.3.3.3 Communications Unit (denial of service)	**4**
		Prevent data transmission	R_O=R4	5	9.3.1.1 Wireless Communications (jamming)	**5**
					9.3.1.2 Backbone Bus (jamming)	4
					9.3.1.3 Chassis Safety Bus (jamming)	4
		Force brake controller into fallback mode	R_O=R4	5	9.3.2.2 ABS and ESP Sensors (disable)	**5**
					9.3.2.1 Chassis Safety Bus (jamming)	4
9.2 Degrade active braking (e.g. by *z* m/s^2)	S_S=0 S_P=0 S_F=0 S_O=2	Manipulate environment information	R_O=R4	5	9.2.1.1Environment Sensors (corrupt)	**5**
					9.2.1.2 Sensor Environment (fake conditions)	**5**
					9.2.1.3 Chassis Safety Bus (insert fake environment data)	1

C.2.3 Tamper with warning message

Tampering with warning messages relates to the use cases "Local Danger Warning to/from other Cars". The risk analysis table (see Table 14) is based on the corresponding attack tree (Figure 8).

Loss of the danger warning function is not expected to result in an additional safety hazard, since it is assumed that drivers will be able to respond to driving hazards by conventional manual braking. Loss of warning messages may not be discernible to drivers, suggesting an operational severity rating S_O=2. However, widespread late or erroneous messages will be more obvious to drivers and are likely to be detrimental to the reputation of this function (and thereby to vehicle manufacturers and system suppliers), suggesting an operational severity rating S_O=3 for the attack objectives "delayed warning" and "wrong warning". No financial or privacy aspects are expected to be associated with this attack.

Table 14 Risk analysis for "Tamper with Warning Message"

Attack Objective	Severity (S)	Attack Method	Risk level (R)	Combined attack probability (*A*)	Asset (attack)	Attack probability (*P*)
5.1 Delay warning message	S_S=0 S_P=0 S_F=0 S_O=3	Delay computation	R_O=R4	4	9.1.1.2 Chassis Safety Controller (denial of service)	2
					9.1.1.1 Communications Unit (denial of service)	**4**
		Delay data transmission	R_O=R5	5	9.1.2.1 Wireless Communications (jamming)	**5**
					9.1.2.2 Backbone Bus (jamming)	4
					9.1.2.3 Chassis Safety Bus (jamming)	4
5.2 Prevent warning message	S_S=0 S_P=0 S_F=0 S_O=2	Prevent computation	R_O=R3	4	9.3.3.1 Chassis Safety Controller (denial of service)	2
					9.3.3.2 Chassis Safety Controller (corrupt code or data)	1
					9.3.3.3 Communications Unit (denial of service)	**4**
		Prevent data transmission	R_O=R4	5	9.3.1.1 Wireless Communications (jamming)	**5**
					9.3.1.2 Backbone Bus (jamming)	4
					9.3.1.3 Chassis Safety Bus (jamming)	4
		Spoof current GPS position	R_O=R3	4	5.2.1 GPS (spoofing)	4
5.3 Display wrong warning message	S_S=0 S_P=0 S_F=0 S_O=3	Tamper with air communications	R_O=R2	2	5.3.2.1 Wireless Communications (listen, intercept, alter, inject, replay)	2
		Tamper with bus communications	R_O=R2	2	5.3.1.1 Backbone bus Communications (listen, intercept, alter, inject, replay)	2
		Fake air communications warning	R_O=R4	4	5.3.3.2 GPS (spoofing) & 5.3.3.1 Wireless Communications (corrupt or fake warning messages)	**4** *5*
		Fake visual warning info	R_O=R1	1	5.3.4.1.1.1 Head Unit (gain root access to embedded OS)	**1**
					5.3.4.1.1.2 Head Unit (malware delivered during flashing)	**1**

C.2.4 Attacking E-Call

This attack relates to the use case "E-Call". The risk analysis table (see Table 15) is based on the corresponding attack tree (Figure 9).

Although this attack does not directly produce safety hazards, there are potential indirect safety implications, since denial or degradation of service may lead to a more severe medical outcome for some casualties. Triggering spurious calls is also an approach to denying or degrading the service. Consequently, the operational severity is set to S_O=3 (significant impact for some, noticeable impact for many) for all attack objectives. No financial or privacy aspects are associated with this attack.

Table 15 Risk analysis for "Attacking E-Call"

Attack Objective	Severity (S)	Attack Method	Risk level (R)	Combined attack probability (*A*)	Asset (attack)	Attack probability (*P*)
6.1 Trigger spurious E-Call	S_S=0 S_P=0 S_F=0 S_O=3	Generate false emergency brake message	R_O=R4	4	5.3.1.1 Backbone bus Communications (listen, intercept, alter, inject, replay)	2
					5.3.3.2 GPS (spoofing) & 5.3.3.1 Wireless Communications (corrupt or fake warning messages)	**4** *5*
		Generate false e-Call message	R_O=R2	2	6.1.2.1/2 Wireless Communications (listen, intercept, alter, inject, replay)	**2**
6.2 Degrade E-Call service quality	S_S=0 S_P=0 S_F=0 S_O=3	Attack service centre	R_O=R3	3	6.2.3.1 Service Centre (overload)	2
					6.2.3.3 Service Centre interfaces (denial of service)	1
					6.2.3.3 Service Centre Interfaces (exploit interfaces)	**3**
		Corrupt transmitted information	R_O=R2	2	6.2.1.1-3Wireless Communications (listen, intercept, alter, inject, replay)	2
		Corrupt GPS information	R_O=R5	5	6.2.2.1 GPS (jamming)	**5**
					6.2.2.2 GPS (spoofing)	4
6.3 Denial of service For E-Call	S_S=0 S_P=0 S_F=0 S_O=3	Attack service centre	R_O=R3	3	6.3.3.1 Service Centre (overload)	2
					6.3.3.2 Service Centre interfaces (denial of service)	1
					6.3.3.3 Service Centre Interfaces (exploit interfaces)	**3**
		Attack communications with service centre	R_O=R5	5	6.3.2.5 Communications Unit (denial of service)	4
					6.3.2.4 Wireless Communications (jamming)	**5**
					6.3.2.1 Service Centre (overload)	2
					6.3.2.2 Service Centre interfaces (denial of service)	1
					6.3.2.2 Service Centre Interfaces (exploit interfaces)	3
		Make Communications Unit contact non-working service centre	R_O=R2	2	6.3.3.1 Communications Unit (corrupt data)	2

C.2.5 Unauthorized brake

The unauthorized brake attack is derived from the use case "Safety Reaction: Active Brake". The risk analysis table (see Table 16) is based on the corresponding attack tree (Figure 11).

The operational impact for unauthorized braking is S_O=4 (significant impact for multiple vehicles). There may also be significant safety implications (S_S=4), potentially with poor controllability (*C3*). The safety-related risks are assessed using the appropriate risk graph (see Table 11). No financial or privacy aspects are associated with this attack.

Table 16 Risk analysis for "Unauthorized brake"

Attack Objective	Severity (S)	Attack Method	Risk level (R)	Combined attack probability (*A*)	Asset (attack)	Attack probability (*P*)
8 False brake command on Chassis Safety Bus	S_S=4, C3 S_P=0 S_F=0 S_O=4	Corrupt environment sensors	R_S=R4 R_O=R2	1	8.3.1 Environment Sensors (flash malicious code to firmware)	1
		Corrupt Chassis Safety Controller	R_S=R5 R_O=R3	2	8.1.1 Chassis Safety Controller (exploit implementation flaws)	**2**
					8.1.2 Chassis Safety Controller (flash malicious to firmware)	1
					8.1.1.1.1 Chassis Safety Bus (listen, intercept, alter, inject, replay)	1
		Spoof brake event in immediate locality	R_S=R7 R_O=R5	4	8.2.1.2.1 Bus Communications (corrupt or fake brake warning message)	1
					8.2.1.1.1.1.1 Head Unit (gain root access to embedded OS) & 8.2.1.1.1.1.2 In-car Interfaces (access)	1 *3*
					8.2.2.2 Wireless Communications (corrupt or fake brake warning message) & 8.2.2.1.1.1 Head Unit (gain root access to embedded OS)	*5* 1
					8.2.2.2 Wireless Communications (corrupt or fake brake warning message) & 8.2.2.1.1.1 Head Unit (exploit vulnerability or implementation error)	*5* 1
					8.2.1.3.1 Domain Controllers (malware delivered during flashing)	3
					8.2.2.2 Wireless Communications (corrupt or fake brake warning message) & 8.2.2.1.1.1 GPS (spoofing)	*5* **4**

C.2.6 Attack E-Toll

The four attack trees relating to the use case "E-Tolling" are combined into a single risk analysis table, since the individual attack trees are more readily associated with attack objectives with implications for the stakeholders. The risk analysis table (see Table 17) is based on the corresponding attack trees (see Figure 13 to Figure 16).

The financial severity of modifying toll payments is perhaps S_F=2 (multiple losses ~€10 perhaps), but the operational severity is probably S_O=3 (noticeable impact for many vehicles) for increased toll payments. Access to private data may allow driver/vehicle tracking of multiple vehicles (S_P=4), but the victims are unlikely to be aware of this (S_O=2). No safety aspects are associated with these attacks.

Table 17 Risk analysis for "Attacking E-Toll"

Attack Objective	Severity (S)	Attack Method	Risk level (R)	Combined attack probability (*A*)	Asset (attack)	Attack probability (*P*)
10 Prevent victim from passing	S_S=0 S_P=0 S_F=0 S_O=3	Attack non-repudiation of billing message	R_O=R2	2	10.1.1.1 Communications Unit (exploit vulnerability of external communication protocols)	**2**
					10.1.1.2.1 In-car ECU (malware flashed with OBD update)	1
					10.1.1.2.2 Communications Unit (malware delivered by mobile device)	**2**
11 Increase victim toll payment	S_S=0 S_P=0 S_F=2 S_O=3	Modify GPS billing data	R_F=R1 R_O=R2	2	11.1.1.1 Communications Unit (exploit vulnerability of external communication protocols)	**2**
					11.1.1.2.1 In-car ECU (malware flashed with OBD update)	1
					11.1.1.2.2 Communications Unit (malware delivered by mobile device)	**2**
12 Reduce own toll payment	S_S=0 S_P=0 S_F=2 S_O=0	Replace GPS trace	R_F=R1	2	12.1.1.1 In-car ECU (malware flashed with OBD update)	1
					12.1.1.2 Communications Unit (malware delivered by mobile device)	**2**
		Fake GPS trace for lower bill	R_F=R4	5	12.2.1 GPS (spoofing)	4
					12.2.2 In car sensors (manipulate heading, speed, orientation)	**5**
13 Access victim's private data	S_S=0 S_P=4 S_F=0 S_O=2	Access user's GPS trace	R_P=R3 R_O=R1	2	13.1.1.1.1 In-car ECU (malware flashed with OBD update)	1
					13.1.1.2.1 Communications Unit (malware delivered by mobile device)	**2**

C.2.7 Green light ahead of attacker

This attack is related to the use cases "Traffic Information to/from Other Entities". The risk analysis table (see Table 18) is based on the corresponding attack tree (Figure 4).

No privacy issues are expected to arise from this attack. However, there may be potential safety hazards from manipulating speed limits or speed limit information; rapid and haphazard changes, for example, may lead to widespread confusion and driver distraction, with potential for multiple minor incidents (S_S=2). Nonetheless, controllability is likely to be reasonable (*C2*) in the urban environments where this type of attack is most likely to be deployed. Possible financial implications could be speeding fines for minor speeding by misinformed vehicles (S_F=2). This could lead to loss of confidence in the systems (S_O=4). No financial or privacy aspects are associated with this attack.

Table 18 Risk analysis for "Green light ahead of attacker"

Attack Objective	Severity (S)	Attack Method	Risk level (R)	Combined attack probability (*A*)	Asset (attack)	Attack probability (*P*)
1.2 Tamper with road-side equip-ment	S_S=2, C2 S_P=0 S_F=0 S_O=4	*Physical modification*		***Not in scope***	*1.2.1.1 Roadside units (modify)*	***Not in scope***
		Take control of roadside units using C2I	R_S=R1 R_O=R2	1	1.2.2.2 Roadside Units (exploit configuration errors) & 1.2.2.1 Roadside Units (exploit protocol implementation flaws)	1 1
1.1 Imper-sonate emergency vehicle	S_S=2, C2 S_P=0 S_F=0 S_O=4	Wireless con-trol of road-side equip-ment	R_S=R2 R_O=R3	2	1.1.1.2 Wireless Communica-tions (fake messages) & 1.1.1.1.1 Wireless Communica-tions (listen, intercept, alter, inject, replay)	*5* **2**
					1.1.1.2 Wireless Communica-tions (fake messages) & 1.1.1.1.2.1.1 In-car ECU or bus (exploit vulnerability or imple-mentation flaw)	*5* 1
					1.1.1.2 Wireless Communica-tions (fake messages) & 1.1.1.1.2.1.2 In-car Communi-cations (listen, intercept, alter, inject, replay)	*5* 1
					1.1.1.2 Wireless Communica-tions (fake messages) & 1.1.1.1.2.2.1.1 Keys (illegal acquisition, modification or breaking) & *1.1.1.1.2.2.1.2.1 Roadside Units (modify)*	*5* 1 *Not in scope*
					1.1.1.2 Wireless Communica-tions (fake messages) & 1.1.1.1.2.2.1.1 Keys (illegal acquisition, modification or breaking) & 1.1.1.1.2.2.1.2.2 Roadside Units (exploit protocol implementa-tion flaws)	*5* 1 1
					1.1.1.2 Wireless Communica-tions (fake messages) & 1.1.1.1.2.3.1 In-car interfaces (physical tampering)	*5* **2**
					1.1.1.2 Wireless Communica-tions (fake messages) & 1.1.1.1.2.3.2 Communications Unit (exploit vulnerability or implementation error)	*5* **2**
					1.1.1.2 Wireless Communica-tions (fake messages) & 1.1.1.1.2.3.3 Head Unit (exploit vulnerability or implementation error)	*5* 1

C.2.8 Manipulate speed limits

This attack is related to the use cases "Traffic Information to/from Other Entities". The risk analysis table (see Table 19) is based on the corresponding attack tree (Figure 5).

No privacy issues are expected to arise from this attack. However, there may be potential safety hazards from manipulating speed limits or speed limit information; rapid and haphazard changes, for example, may lead to widespread confusion and driver distraction, with potential for multiple minor incidents (S_S=2). Nonetheless, controllability is likely to be reasonable (*C2*) in the urban environments where this type of attack is most likely to be deployed. Possible financial implications could be speeding fines for minor speeding by misinformed vehicles (S_F=2). This could lead to loss of confidence in the systems (S_O=4). Such an attack could also be part of a wider objective to manipulate traffic flow or to cause a traffic jam.

Alternatively, an attacker may simply be trying to exceed authorised speed limits. Excessive speed in the urban environment may also pose an increased safety hazard for pedestrians and other road users (S_S=3), with poor controllability (*C3*). A possible motivation for this could be to avoid fines for significant speeding, so there may also be a financial implication for the authorities (perhaps S_F=3). Reducing speed limits could also be part of a wider objective to manipulate traffic flow or to cause a traffic jam.

Table 19 Risk analysis for "Manipulate speed limits"

Attack Objective	Severity (S)	Attack Method	Risk level (R)	Combined attack probability (*A*)	Asset (attack)	Attack probability (*P*)
2.2 Modify limits enforced by roadside equipment	S_S=3, C3 S_P=0 S_F=3 S_O=0	*Fake wired speed update messages from authority*		***Not in scope***	*2.2.2.1 Wired infrastructure (fake speed limit messages)*	***Not in scope***
		Take control of roadside units	R_S=R3 R_F=R1	1	2.2.1.3 Roadside Units (exploit configuration errors)	**1**
					2.2.1.4 Roadside Units (gain root access)	**1**
					2.2.1.2 Roadside Units (exploit protocol implementation flaws)	**1**
					2.2.2.1 *Roadside Unit to Authority communication (access wire infrastructure)*	***Not in scope***
					2.2.1.1 Roadside Units (modify)	***Not in scope***
2.1 Issue bogus speed limit notices to other vehicles	S_S=2, C2 S_P=0 S_F=2 S_O=4	Impersonate authority	R_S=R2 R_F=R1 R_O=R3	2	2.1.2.1 Wireless Communications (replay speed limit message)	2
					2.1.2.2 Authorisation keys (illegal acquisition by physical attack)	***Not in scope***
		Influence roadside equipment	R_S=R5 R_F=R4 R_O=R6	5	2.1.3.1 Wireless Communications (fake traffic conditions messages)	5
		Take control of roadside units	R_S=R1 R_F=R0 R_O=R2	1	2.1.1.2 Roadside Units (exploit configuration errors)	**1**
					2.1.1.3 Roadside Units (exploit protocol implementation flaws)	**1**
					2.1.1.4 Roadside Units (gain root access)	**1**
					2.1.1.1 Roadside Units (modify)	***Not in scope***

C.2.9 Simulate traffic jam

This attack is related to the use cases “Traffic Information to/from Other Entities”. The risk analysis table (see Table 20) is based on the corresponding attack tree (Figure 7).

No direct safety or privacy implications are expected to arise from this attack. However, there may be operational threats in terms of customer dissatisfaction and loss of reputation for vehicle manufacturers and their system suppliers (S_O=4), as well transport authorities and ITS system operators. Possible financial implications could include loss of earnings for individuals and more widespread harm to the economy (S_F=4).

C.2.10 Manipulate traffic flow

This attack is related to the attack trees “unauthorized braking”, “green light ahead of attacker”, “simulate traffic jam” and “manipulate speed limits”. The risk analysis table (see Table 21) is based on the corresponding attack tree (Figure 6).

No privacy issues are expected to arise from this attack. However, there may be potential safety hazards from manipulating speed limits or speed limit information, or from routing traffic the wrong way into one-way systems, with potential for multiple minor incidents. (S_S=2). More serious is the possibility of exploiting the unauthorized brake attack as a means of slowing or stopping cars. The operational impact for unauthorized braking is S_O=4 (significant impact for multiple vehicles), and there may also be significant safety implications (S_S=4) associated with such attacks. Nonetheless, controllability of safety hazards is likely to be reasonable (*C2*) in the urban environments where this type of attack would be most likely to be deployed.

Possible financial implications could be fines for minor speeding or other breaches of traffic regulation by misinformed vehicles (S_F=2). The financial implications associated with inducing traffic jams could be more severe (S_F=4), including loss of earnings for individuals and more widespread harm to the economy. Such attacks could also lead to loss of confidence in the associated systems if security breaches are identified as the mechanism (S_O=4).

Table 20 Risk analysis for "Simulate traffic jam for target car"

Attack Objective	Severity (S)	Attack Method	Risk level (R)	Combined attack probability (*A*)	Asset (attack)	Attack probability (*P*)
4.1 Attack I2C messages	S_S=0 S_P=0 S_F=4 S_O=4	Impersonate roadside equipment	R_F=R3 R_O=R3	2	4.1.2 Wireless Communications (insert fake RSU traffic jam warning)	2
		Forward traffic jam message and falsify location	R_F=R3 R_O=R3	2	4.1.1.2 Wireless Communications (modify position data in traffic jam message from other location) & 4.1.1.1 Wireless Communications (forward modified traffic jam message)	2 2
4.3 Attack C2I messages	S_S=0 S_P=0 S_F=4 S_O=4	Alter position data for cars in traffic jam	R_F=R5 R_O=R5	4	4.3.2.1.1.1 Wireless Communications (listen, intercept, alter, inject, replay)	2
					4.3.1.1.1 Wireless Communications (corrupt or fake messages & 4.3.2.1.1.1 Wireless Communications (listen, intercept, alter, inject, replay	*5* 2
					4.3.1.1.1 Wireless Communications (corrupt or fake messages & 4.3.2.1.1.2 In-car sensors (spoofing)	*5* 2
					4.3.1.1.1 Wireless Communications (corrupt or fake messages & 4.3.3.2.1 GPS (spoofing)	*5* **4**
					4.3.2.1.1.1 Wireless Communications (listen, intercept, alter, inject, replay) & 4.3.2.1.1.2 In-car sensors (spoofing)	2 2
					4.3.2.1.1.1 Wireless Communications (listen, intercept, alter, inject, replay) & 4.3.3.2.1 GPS (spoofing)	2 *4*
		Simulate the existence of many cars	R_F=R6 R_O=R6	5	4.3.1.1.2 Wireless Communications (exploit vulnerability or implementation errors)	2
					4.3.1.1.1 Wireless Communications (corrupt or fake messages)	**5**
		Modify car to believe itself in a traffic jam	R_F=R5 R_O=R5	4	4.3.3.2.2 In-car ECU or bus (configuration changes)	2
					4.3.3.2.1 GPS (spoofing)	**4**
4.2 Tamper with roadside equipment	***Not in scope***	*Attack infrastructure*			*4.2.1 Roadside Units (exploit protocol implementation flaws)*	*Not in scope*
					4.2.2 Roadside Units (exploit configuration errors)	

Table 21 Risk analysis for "Manipulate traffic flow"

Attack Objective	Severity (S)	Attack Method	Risk level (R)	Combined attack probability (*A*)	Asset (attack)	Attack probability (*P*)
Divert vehicles	S_S=2, C2 S_P=0 S_F=2 S_O=4	Spread bogus accident warnings	R_S=R2 R_F=R1 R_O=R3	2	*3.1.1.1.1 Roadside Unit (access)*	*Not in scope*
					3.1.1.1.2.1.1 In-car Communications (listen, intercept, alter, inject, replay)	1
					3.1.1.1.2.1.2 Keys (illegal acquisition, modification or breaking)	1
					3.1.1.2.1.1 Wireless Communications (exploit vulnerability or implementation error)	**2**
					3.1.1.2.1.2 Head Unit (exploit vulnerability or implementation error)	1
					3.1.1.2.1.3 Communications Unit (exploit vulnerability or implementation error)	**2**
					3.1.1.2.2.1 Wireless Communications (listen, intercept, alter, inject, replay)	**2**
		Spread bogus traffic jam information	R_S=R5 R_F=R4 R_O=R6	5	3.1.2.1.1 Wireless Communications (listen, intercept, alter, inject, replay)	2
					3.1.2.1.3 Wireless Communications (exploit vulnerability or implementation error)	2
					3.1.2.1.3 Wireless Communications (corrupt or fake messages)	**5**
Induce traffic jam	S_S=4, C2 S_P=0 S_F=4 S_O=4	Make cars stop (see unauthorized brake attack)	R_S=R6 R_F=R5 R_O=R5	4	8.3.1 Environment Sensors (flash malicious code to firmware)	1
					8.1.1.2 Chassis Safety Controller (exploit implementation flaws)	2
					8.1.1.2 Chassis Safety Controller (corrupt code or data)	1
					8.2.1.1.1.1.1 Head Unit (gain root access to embedded OS) & 8.2.1.1.1.1.2 In-car Interfaces (access)	1 *3*
					8.2.1.2.1 Bus Communications (corrupt or fake brake warning message)	1
					8.2.2.2 Wireless Communications (corrupt or fake brake warning message) & 8.2.2.1.1.1 Head Unit (gain root access to OS)	*5* 1
					8.2.2.2 Wireless Communications (corrupt or fake brake warning message) & 8.2.2.1.1.1 Head Unit (exploit vulnerability or implementation error)	*5* 1

					8.2.1.3.1 Domain Controllers (malware delivered during flashing)	3
					8.2.2.2 Wireless Communications (corrupt or fake brake warning message) & 8.2.2.1.1.1 GPS (spoofing)	5 **4**
		Slow cars down (includes manipulate speed limits attack)	R_S=R7 R_F=R6 R_O=R6	5	3.2.2.1.1 Wireless Communications (corrupt or fake messages)	**5**
					3.2.2.1.2 Wireless Communications (listen, intercept, alter, inject, replay)	2
					3.2.2.1.3 In-car Communications (listen, intercept, alter, inject, replay)	1
					3.2.2.3.1.1.2 In-car ECU or Bus (exploit vulnerability or implementation error)	1
					3.2.2.3.3.2.1 Head Unit (exploit vulnerability or implementation error)	1
					3.2.2.4.1.1/2 In-car Interfaces (access – exploit vulnerabilities)	3
					3.2.2.4.2.1 In-car Communications (corrupt or fake messages)	2
					3.2.2.4.2.2 In-car sensors (spoof)	2
					3.2.2.4.2.3 In-car sensors (manipulate)	**5**
					2.1.1.3 Roadside Units (exploit configuration errors)	1
					2.1.1.2 Roadside Units (exploit protocol implementation flaws)	1
					2.1.2.1 Wireless Communications (replay speed limit message)	2
					2.1.3.1 Wireless Communications (fake traffic conditions messages)	**5**
					2.1.1.3 Roadside Units (exploit configuration errors)	1
					2.1.1.3 Roadside Units (exploit protocol implementation flaws)	1

C.2.11Engine denial of service

This attack is not directly related to any of the use cases, but many of them may provide opportunities for mounting attacks (i.e. those involving wireless communications, nomadic devices and workshop updating). The risk analysis table (see Table 22) is based on the corresponding attack tree (Figure 10).

No direct safety, privacy or financial implications are expected to arise from this attack. However, there may be operational threats in terms of customer dissatisfaction and loss of reputation for vehicle manufacturers and their system suppliers (S_O=3).

Table 22 Risk analysis for "Engine denial of service"

Attack Objective	Severity (S)	Attack Method	Risk level (R)	Combined attack probability (*A*)	Asset (attack)	Attack probability (*P*)
7.1 Engine controller is not reachable	S_S=0 S_P=0 S_F=0 S_O=3	Disable Powertrain Controller	R_O=R2	2	7.1.2.1 Powertrain Controller (PTC malware flashed with OBD update)	1
					7.1.1.2 In-car ECU or bus (configuration change – bus parameters changed to disable)	**2**
		Powertrain Controller unreachable	R_O=R4	4	7.1.3 In-car Communications (jamming)	4
		Disable Engine Control Unit	R_O=R2	2	7.1.4 In-car ECU (disable or denial of service)	2
					7.1.4.1 Engine Control Unit (ECU malware flashed with OBD update)	1
7.4 Powertrain controller does not receive order	S_S=0 S_P=0 S_F=0 S_O=3	Owner access denied	R_O=R4	4	7.4.4.2 Cryptographic Data (modified)	1
					7.4.4.1 In-car Communications (jamming)	**4**
		Message corrupted	R_O=R2	2	7.4.1 In-car Communications (corrupt message or data)	2
		Backbone Bus disabled	R_O=R4	4	7.4.3 Backbone Bus (denial of service)	4
7.2 Engine controller receives warning	S_S=0 S_P=0 S_F=0 S_O=3	False warning from Powertrain Controller	R_O=R2	2	7.2.2.2 Powertrain Controller (corrupt code or data)	1
					7.2.2.3.1.2 Communications Unit (malware flashed with OBD update)	1
					7.2.2.3.1.1 Wireless Communications (exploit vulnerability of external communication protocols)	**2**
		False warning on Powertrain Domain Bus	R_O=R1	1	7.2.1 Powertrain Peripherals (corrupt code or data)	**1**
					7.2.2.2 Powertrain Controller (PTC malware flashed with OBD update)	**1**
7.3 Essential engine component out of order	S_S=0 S_P=0 S_F=0 S_O=3	Previously issue bogus commands to induce damage	R_O=R2	3	7.3.1.1 In-car Communications (corrupt or fake messages on Powertrain Bus)	2
					7.3.1.2 In-car Communications (exploit vulnerability of external communication protocols)	**3**
					7.3.1.3 In-car ECU or bus (exploit vulnerability or implementation error)	1

C.3 Summary and conclusions

Analysis of the attack trees demonstrates that specific asset attacks may contribute to different attack objectives within the same attack tree, and may also contribute to attack objectives associated with other attack trees. For a particular asset attack, both the risk level (reflecting the attack potential and the severity of outcome) and the number of instances from the collec-

tion of attack trees are indicators of the importance of the asset attack and the likely benefits of measures for reducing its attack potential.

The attack potentials for jamming GPS and wireless transmissions are very high and dominate several of the risk analyses. Measures for detecting tampering with GPS and wireless transmissions, and avoiding reliance on GPS signals alone as a source of position data, would help to reduce the associated risk levels.

It should be noted that EVITA is not aiming to develop new vehicle systems with particular levels of security, or to enhance the security of existing systems. The EVITA project is concerned with prototyping a "toolkit" of security measures (which may software, hardware, and architectural) that could be selected for further development and implementation in future systems. Consequently, the requirements analysis activity is based on a representative range of possible applications and a generic vehicle architecture, with the aim of identifying what kind of security requirements may need to be met, as well as their likely prevalence and distribution amongst the system assets.

Nonetheless, it is expected that much of the EVITA security engineering process could be adapted to support future product development processes. The risk analysis approach could be used, in combination with the vehicle manufacturer's security policy, in order to decide whether to accept or transfer the identified security risks, or to take measures to reduce or avoid specific risks where this is deemed necessary.

Appendix D – Identifying security requirements

D.1 Abstract functional path approach

D.1.1 Abstract functional system model

As a basis for the security requirements analysis, a functional model is derived from the use cases. The nature of the use case descriptions is such that it is not possible to identify the complete system under investigation. Therefore an abstract functional component model is developed, which represents the behaviour of a single car within the system (see Figure 19). It provides an overview of every action happening at the functional borders of the car component, as well as the interactions with other cars or with other entities of the system.

The functional flow is illustrated in the form of arrows from inputs[4] to outputs[5] of the system component model within the TOE box (Target of Evaluation). The arrows outside the TOE box denote the functional relations between different components of the global system. The intermediate predicates[6] represent an abstraction of the condition checking, which leads to a certain system action. It is not intended to represent the behaviour of the system component under investigation, but only the implicit functional decisions. The predicates are not defined further, as this is the subject of research being done in the area of safety reaction systems and not directly related to security. Also, it is not necessary to define them in our abstract model in order to derive the necessary security requirements, aside from ensuring a guaranteed behaviour of the system.

Based on the functional component model, one may now start to reason about the overall system. The synthesis of the inner and the outer system behaviour builds the global system behaviour. The distinction between internal and external flow description regarding the functional component model is expressed in terms of internal and external functional flow.

[4] Inputs: DSRC-Receive Neighbourhood-Information), DSRC-Receive(C2X-Message(Emergency)), Environment-sensing(Environment-Information), Chassis-Sensing(Vehicle-Dynamics), DSRC-Receive(Cooperative-Awareness-Message), DSRC-Receive(Traffic-Information-Message), Sensing(Data), GPS-Sensing(Position), BT-Receive(OpenHood), HMI-Read(POI-Configuration), Receive(POI-Info), USB-Receive(Software), HIM-Read(Inputs), BT-Receive(Display(Data)), HIM-Read(Inputs), BT-Receive(SeatPosition), Receive(Diagnosis-Request), DSRC-Receive(Firmware), Diag-Receive(Firmware)

[5] Outputs: DSRC-Send(Neighbourhood-Token), DSRC-Forward(C2X-Message(Emergency)), Brake, Driving-Power-Reduction, DSRC-Send(C2X-Message(Emergency)), HMI-Display(Warning), HMI/Navigation-Display(Warning), PTC-Action, DSRC-Send(Cooperative-Awareness-Message), Send(Traffic-Information-Message), GSM-Send(Billing-Information), GSM-Send(eCall-Request(Position), Open(Hood), HMI-Show(POI-Info), HIM-Show(SW-Interface), HIM-Show(Data), BT-Send(Inputs), Adjust(SeatPosition), Send(Diagnosis-Data)

[6] Intermediate predicates: Forwarding-Message=true, Danger-Avoidance->Emergency-Braking=true,Processing->Warning=true, Processing->ShowInfo=true, Situation-Assessment->Emergeny=true, Processing->critical-situation-recognition=true, Aggregation, Collecting->TollRoad->Calculation=true, Crash-calculation=true, Show-POI=true, Execution

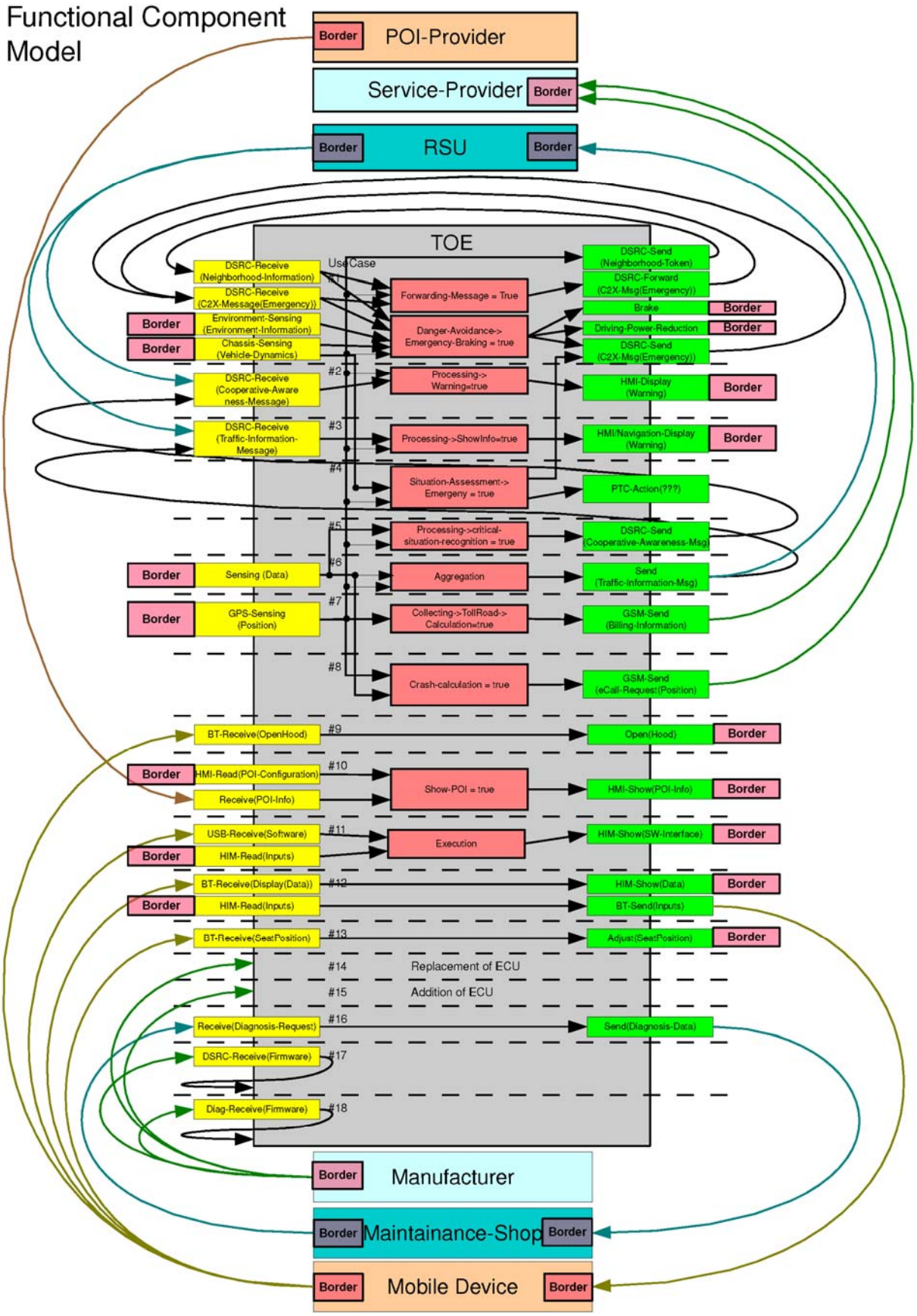


Figure 19 Abstract functional system model pattern

Figure 20 illustrates an example of a functional system model instance derived from the functional component model (Figure 19) with a subset of actions. This should help to illustrate how to interpret the component model. Of course, an exhaustive list of all possible instances of the system models would be too big to be written down. Therefore, the identification of border actions of the overall system that are relevant to the security requirements is performed within the component model. However, the functional dependencies among several component instances must still be taken into account during this process.

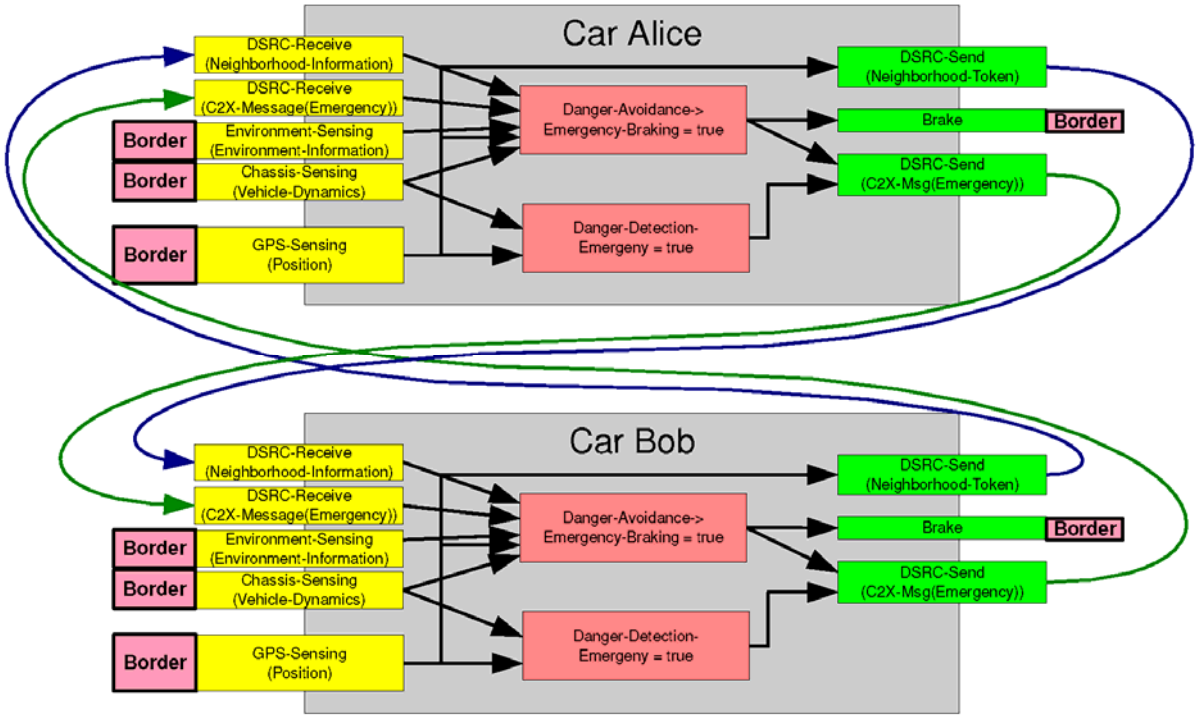


Figure 20 Abstract functional system model instance

D.1.2 Security Requirements Engineering Process

D.1.2.1 Authenticity

Each functional flow arrow within the system description describes a functional dependency in the reverse direction. Associated with each functional dependency is an authenticity requirement. Therefore, each functional flow arrow can be viewed as an authenticity requirement in the reverse direction that spans from the output border action to all input border actions that it depends and relies on.

Additionally, authenticity is required regarding the actions “Replacement/Addition of ECU” and “DSRC / diag receive (firmware)” for every output action performed by the system under investigation. These requirements originate from the system’s dependence on the underlying hardware and software. Accordingly, changes to those may result in an alteration of the intended system behaviour. Also for the driver of a following car, or an RSU, this must be authentic.

As the critical point regarding liability and safety is the driver, we assume that the driver is the subject of each of the authenticity requirements. This means, that the driver is the agent that must be assured of all requirements regarding the functional safety of the car2car system.

D.1.2.2 Confidentiality

For confidentiality we will assume a strategy of minimal disclosure: Unless the functional flow requires a disclosure of information, the information shall be confidential. These requirements can then be grouped regarding the kind of data they address regarding classical confidentiality, anonymity and privacy.

D.2 Detailed functional path and mapping approach

D.2.1 Methodology

D.2.1.1 Introduction

Developing a model of the system to be analysed usually requires that it already exists or is fully specified. In the context of the security requirements definition things are slightly different because the purpose is to *contribute* to the specification process by adding some security related constraints. We thus need to live with a system that is neither yet available nor specified. The following is an attempt to define a model in a generic way, using a set of generic physical and functional components, characterized by a set of generic parameters. Our primary goal is to offer a framework dedicated to security analysis, flexible enough to be adapted to any actual system but still capable of producing accurate analysis results. This framework could be used as follows:

1. A candidate on-board network structure is designed by selecting components, interconnecting them and characterizing them (that is, assigning values to their generic parameters).
2. One or several typical use cases are selected as a starting point of investigation.
3. The functional description of the use cases is built as a graph of communicating tasks. As for the architecture this is done by selecting, connecting and characterizing generic components.
4. The functional description is mapped on the architecture: each task is allocated to a computing node; each logical communication channel is allocated to physical links and memories. All the mapping parameters are set (arbitration policies, priorities, etc.) This defines a fully mapped system.
5. Attack trees are designed representing the different ways an attacker could pursue his/her goals. The attack trees are refined up to the point where more information is needed about the architecture (internals of a component) or a branch can be cut thanks to the parameters of the mapped system (the cost/skills factor exceeds the threshold for the considered goal/attacker's class). Attack tree nodes are fully numbered so as to further evaluate the coverage of attacks.

6. The architecture, the functional description and the mapping are refined whenever needed to further explore a branch.
7. Remaining branches are cut by modifying the system (architecture and/or functional description and/or mapping). New attacks may also have been identified
8. Security requirements are listed. They are found considering use cases' description, functional and mapping views, as well as attack trees. Those requirements shall provide a full coverage of attack tree nodes.
9. Attack trees might be updated according to new attacks that may have been found when modelling functional and mapping views, or when listing requirements. Then, we restart from point "5." until the list of requirements remains unchanged.

Risk evaluation of the system under investigation (every remaining branch in an attack tree represents a risk) is also an output of that methodology. The graph shown in Figure 21 illustrates the overall analysis flow.

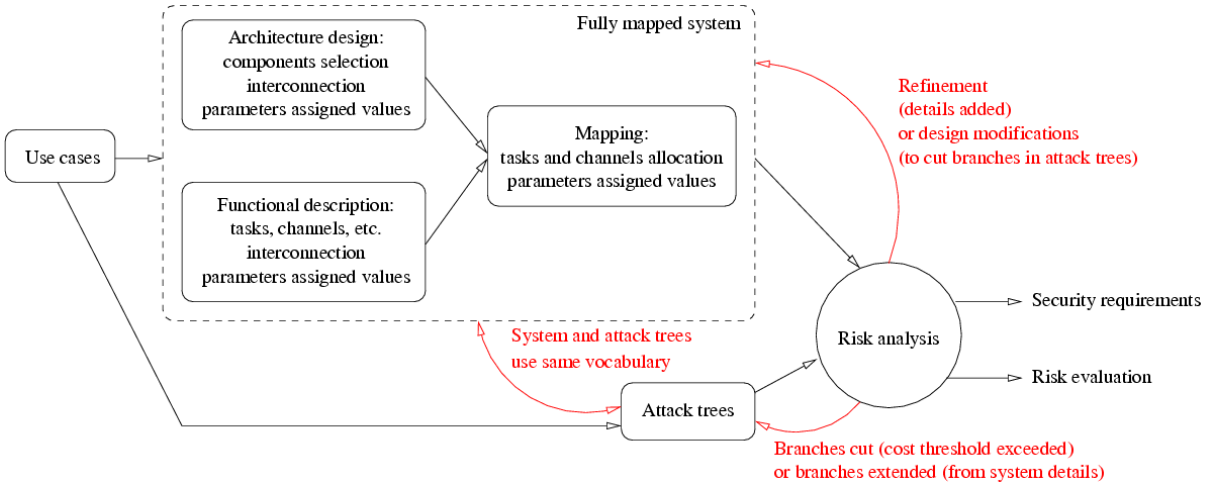


Figure 21 Identification of security requirements

D.2.1.2 Functional and architectural description

A full system is made of a functional specification, a physical architecture and a mapping of the former to the latter.

A functional description is a collection of communicating functions. In the DIPLODOCUS approach, functions may communicate using either abstract data channel, or abstract event. This difference of communication semantics is not really meaningful for capturing security requirements. Also:

- Functions generating input to the system are considered as related to sensors.
- Functions generating commands to the outside of the system are related to actuators
- Other functions are meant to be executed on either CPU nodes, or hardware nodes such as hardware accelerators or I/O devices.

Note that, even if functions are meant to be mapped on physical architectures, the functional specification shall describe the system with as few references to an underlying physical architecture as possible.

An architectural description is a collection of interconnected hardware nodes. Those hardware nodes are computing nodes (CPUs, I/O devices, hardware accelerators), storage nodes (RAM, etc.), sensors and actuators. The interconnection between those nodes is described in term of busses, networks and wireless links.

The general mapping framework is the following. Each functional element of the system under investigation is mapped on a physical component of the architectural view. Tasks are mapped on computing nodes and on memories. A task has code memory segments and data memory segments. Each memory segment must be mapped on a memory component or on a computing component (the latter case represents embedded memories that are not yet visible at this architecture refinement level). Data segments cannot be mapped on read only memory components (read-only data are considered as a code segment). Segments are optionally split in two views: the load view and the runtime view. When a code segment is split, its two views can be mapped on two different memory components. During the initialization of the system the load view is read from its memory and copied in another, creating the runtime segment. An un-split code segment and the load view of a split code segment cannot be mapped on a volatile memory. The load view of a split data segment cannot be mapped on a volatile memory. An unsplit data segment can be mapped on a volatile memory (in this case it is uninitialized or always initialized to the same value at boot time).

The systems under consideration carry out computing and communicating activities. Both activities usually rely on a third important one: information storage. Acquisition of environment characteristics and physical actions on mechanical devices are the two other important activities of automotive systems taken into account, both at security requirement level and at modelling level. Note that, from a functional point of view, communication and storage look very similar (sending and writing could be considered as the same operation; receiving and reading too), but they are different: Communication takes place between different tasks while storage is dedicated to a single task, for its own needs. Moreover, reading is an action while receiving a message is an event: A task decides to read or not but has no control on messages reception, even if received messages can be ignored. Of course, when considering the physical view, it may be that communications are implemented through read and write operations in a memory and, in most cases, read and write operations of a task are implemented as transactions on a physical communication link between a processor and its external memories. Each of the five activities has a physical and functional counterpart, as outlined in Table 22.

Table 23 Views on activities

Activity	**Physical view**	**Functional view**
Computing	CPU or dedicated hardware accelerator	Processing task
Communication	Wired bus or network, wireless link	Send/receive messages on logical channels
Storage	Memory (RAM, ROM, flash)	Read/write data from/to address spaces
Acquisition	Sensors	Get measurements from the environment
Command	Actuators	Do actions on the environment

D.2.2 Classification of attacks

We will consider two classes of attacks:

- attacks modifying the behaviour of the system (active attacks) and
- attacks aiming at information retrieval without modifying the behaviour (passive attacks).

Note: passive attacks are frequently a pre-requisite of active attacks; the attacker first analyses the system in a passive way to understand it or recover useful crypto material and then exploits this knowledge to actively attack. There are two possible ways to identify the different types of attacks: the physical one and the functional one. In the following we analyse the attacks against our five physical components. For each physical attack we also indicate which functional attack it can be used for.

Active attacks:

1. Computing attacks are targeting computing nodes (CPUs, hardware accelerators). They consist in physical modifications of the component (like modifying the content of an embedded ROM or the structure of an operator), its replacement or even its destruction. Transient faults injection is another way. The consequence is the production of results that differ in some way from those that would have been produced in normal operations, including failure to produce results when expected or the converse. Note: the purpose of a fault attack is very frequently to retrieve an embedded secret, in which case the modification of the behaviour is not the real goal of the attacker but more something like a mean. On the functional point of view, these physical attacks can translate into computing, communication, storage or command attacks: modifying the behaviour of a task can indeed lead to modifications of the results it produces, of the messages it exchanges with its environment, of the content of the memories it manages or of the orders it sends to actuators.
2. Communication attacks are targeting communication links. There are two main means to implement attacks against a physical communication link: tampering with it (modifying its topology, jamming it, modifying its main parameters like arbitration policy, frequency, etc.) and injection of forged transactions. Because computing devices and memories are usually connected through busses, attacks against communication links can be used to tamper with the communication or the storage activity. Consequences of communication attacks are on the receiver side only (attacks aiming at modifying or cancelling a message before it is actually sent are in fact attacks against the sending computing node). They comprise the modification of a message between its emission and its reception, the cancellation of a message that will thus never reach its destination or the reception of a message that would never have been received during normal operation. When a memory bus is attacked, it can be to modify the function of a task (software code modification) or the data it processes. There are three classes of memory bus injection attacks: *spoofing* (the injected information was forged by the attacker), *splicing* (the injected information was taken at a different location in the memory) and *replay* (the injected information was taken at the same location in the memory but at a previous date, where it differed from the expected one). The same classes apply to messages. The attacks against communication links are the more powerful of all because, on the functional point of view, these physical attacks can translate in computing, communication, storage, acquisition, or command attacks.

3. Active attacks against storage all consist in modifying the regular content of a memory. As a consequence the read operations performed by the tasks accessing the address space do not return the expected information, that is, the last one that was written at the same location. The consequences are very similar to the consequences of attacks against memory busses. The means used to achieve content modification depend on the technology: ROMs can be replaced, non-volatile writeable memories (EEPROMs, flashes) can be replaced or reprogrammed, volatile memories (static and dynamic RAMs) are much more difficult to attack in a conscious way but more or less random bit flips can be induced by voltage, clock frequency, temperature modifications or more active fault attacks. In some cases, volatile memories can even be cooled, removed from their PCB and plugged on another host without losing their content which can then be read out and / or modified before the component is plugged back in its regular host system.
4. The consequences of acquisition attacks are the production of altered measured metrics. They consist in artificial modifications of the environment (like, for instance, the use of a heating device to increase the measured temperature), modification, destruction or replacement of the sensor.
5. Attacks against actuators, as for the attacks against sensors, consist in modifications of the environment (increase of the friction to reduce the effect of an action applied with a constant force), modification, destruction or replacement of the actuator. They lead to a different action than the intended one.

Passive attacks:

1. Passive attacks against the computing activity aim at retrieving either a secret quantum of data (secret key) or the processing definition itself (software code extraction). As every computation is actually performed by a physical device, measurable syndromes are produced, like power consumption, computing time or electro-magnetic emissions that can be exploited to guess what operations are performed or what is the value of some sensitive data. This kind of analysis is referred to as side channel attacks in the literature. Observing the external communication or the exchanges with memories is another mean to get information about the computing but fall in the passive communication attacks category.
2. Communication can be spied at and sensitive messages or read/written data exposed. On-board or on-bus probing is a very effective and attractive mean for wired communications. Wireless communications are even more sensitive to this kind of attack as they can be conducted in a completely remote and undetectable way. On-chip probing requires package removal, expensive equipment and very skilled attackers.
3. Storage passive attack consists in reading the content of a memory. Some very sophisticated analysis tools can be used to investigate memories but they usually imply package removal plus some on-silicon scanning. Memories can also be dumped from their regular I/O. A ROM or a non-volatile memory can be isolated or even removed from its printed circuit board, its address bus driven and its content recorded by a logic analyser or any similar equipment. In some cases, volatile memories can even be cooled, removed from their PCB and plugged on the recording host without losing their content.
4. Passive attacks against sensors are not applicable.

Security requirements listed in that document address both active and passive attacks. Functional and architectural views used for modelling the system capture some of the hardware and software elements mentioned above, such as CPU, RAM, TOM, communication links, etc. Some elements might be captured only at architectural view level.

From a security point of view they are all potential targets of attacks but by different means and consequences. Security requirements shall therefore address all system elements that might be involved in attacks identified in attack trees.

Additionally, attacks might be classified according to the system description itself. To do so, we might provide attributes to functional and physical element, such as the cost to make a software exploit on a given software task, or the cost to dump a given memory. This full set of attributes will be defined and used at formal verification step (Task 3400).

D.2.3 SysML based security requirements

D.2.3.1 Rationale

The increasing complexity of large-scale heterogeneous systems such as embedded systems has made requirements engineering the most critical phase during system conceptualization. Security requirements in particular should be specified and taken into account before the system architecture is fully defined. However, determining such requirements within embedded systems generally and paradoxically necessitates a detailed enough knowledge of the system components and interactions, like how functions are mapped onto hardware, whether some communication might be seen by an attacker, etc. Security requirements in fact constitute the most abstract documentation of the expected system behaviour.

As such, security requirements should provide a specification that has to be satisfied at every subsequent stage of the system design, validation, development, and testing. Establishing relationships between requirements and such later phases of engineering should thus receive appropriate support: for instance, it should be possible to document the fact that some security mechanism is introduced in order to satisfy one security requirement, or to point at some test over the implementation in order to verify that it is compliant with the same requirement. In the case of embedded systems, the need for hardware protection to satisfy security requirements should be supported by the methodology used.

Security requirements should furthermore constitute a manageable documentation for the average system engineer. As of today, requirements are mainly defined using natural language descriptions and are largely text based. However, such techniques are often imprecise and may lead to the specification of inconsistent security requirements. In particular, security requirements are often defined independently from the security threat analysis or, on the contrary, they are mixed together. A precise description of a separately defined threat coverage however is necessary to provide convincing arguments as to the security achieved by the system under design. Furthermore, a text based description also does not make it possible to define relationships between different requirements that would make it possible to organize the set of security requirements into a description with different levels of complexity that would be more manageable by a human being. In that respect, graphical formalisms typically exhibit many advantages in terms of human readability, although text-based tables may be more appropriate for checking the consistency of a set of requirements.

Finally, because they are defined on a partially specified system, security requirements are likely to evolve during the engineering process. Requirement traceability is thus another important issue which is missing in text-based approaches: providing a rationale about the definition of fine-grained requirements is necessary to understand whether some requirement is still necessary if some assumption about the environment, an attacker, or even the system architecture changes. Another problem with tracing the origin of some requirement stems from the fact that system design is generally architecture and function driven, whereas security is generally non-functional and introduced orthogonally to components.

D.2.3.2 Related Work

Modelling and validating requirements is a topic at stake in system engineering [22][23][24] more specifically to propose standardized environments/languages. Many specification approaches have followed the road of defining profiles [22][24][26][27][28][29] based on the Unified Modelling Language (UML) [25]: in addition to the graphical specification, portability and interoperability are usually among the strength of such profiles. The specification also being semi-formal provides a great deal of flexibility at an early stage of engineering. The next paragraphs discuss the ins and outs of several of those profiles.

A UML profile has been introduced based on the KAOS methodology, one of the most advanced approaches to the specification of security requirements. KAOS provides a language and method to goal-driven requirements design [30] yet was not originally devised with a UML centric approach in mind. KAOS provides semantic elements to represent time, agents, events, goals, goal patterns, goal categories and subgoals as well as conflicting goals and constraints. A relevant feature of KAOS is that goals and related constraints can be defined formally using temporal logic [28]. However, in order to define such expressions accurately, design should be as precise as possible. KAOS properties also span several classes and may properly express non-functional requirements like security ones [28]. However KAOS lacks a systematic coverage of threats and does not provide any support for software-hardware co-design nor code generation and testing in that setting.

The Enterprise Distributed Object Computing profile (EDOC) [30] relies on the Object Constraint Language (OCL) to represent and check requirement satisfaction. However, this profile has been designed to specify functional requirements rather than security ones, and also to improve business processes. It thus does not provide appropriate support for security requirements specification.

The Refinement Calculus for Object System (rCOS) is an object-based language with a rich variety of features including subtypes, inheritance, type casting, dynamic binding, and polymorphism [31]. rCOS permits the mathematical characterization of objects through Labelled Transition Systems. Despite its benefits, rCOS is inherently software oriented, which is perfectly fit for service oriented design [26] but not appropriate for the specification of security requirements at system level.

UMLsec [33] introduces a security-oriented methodology based on activity diagrams, state-charts, sequence diagrams, class diagrams, and deployment diagrams. As shown in [34] the UMLsec approach can be complemented with automated verification of security properties using an industrial tool. The examples provided with UMLsec outline that the methodology is more about security mechanism design than about security requirements linked with use cases: guidelines might be provided for handling secrecy, secure key management, and

security protocol specification. In addition, the specification might be ambiguous due to the UML diagrams used: for instance, defining integrity on top of a link between execution nodes in the deployment diagram may be interpreted either as a property that the link should implement or as a mechanism that the link already implements independently from requirements. Another drawback of UMLsec comes from the fact that it is an extension of UML, and therefore not recognized by most existing tools. Some authors [35] also even claimed that OCL constraints were enough to introduce similar specifications without extending UML.

SecureUML is a profile that aims to provide security again using an extension of the UML specification [36]. SecureUML however only specifies security polices for Role-Based Access Control (RBAC) using graphical notation and logical constraints, so it is essentially adapted to application security.

Since these UML profiles are design-oriented, they are generally more suited to describing security mechanisms than security requirements. Other non UML-based environments can also be mentioned, that rely directly on a formal framework like for instance the Symbolic Trace Analyser (STA). STA is a model checker for cryptographic protocols relying on symbolic techniques [37]. Protocols are described in a dialect of the spi calculus. Intruders can be modelled based on the well-known Dolev-Yao model. STA allows to express and verify authentication and secrecy properties. The lack of parametrization of STA leads to large specifications when a more instances are needed. Another example is the On-the-fly Model-Checker (OFMC). OFMC implements bounded verification of protocols by exploring its transition system described in a specification in a demand-driven way. OFMC also support the specification of algebraic properties of cryptographic operators. To our knowledge STA and OFMC, as well as other non UML-based security-oriented environments, do not have specific constructs to represent security properties. An exception to the latest statement could be ST-Tool [23]. ST-Tool provides a Graphical User Interface, a Data Modeller and Formal Language and Analysis components that are based on the formal language TROPOS. This profile is mainly intended for Security Requirements Engineering. A general problem with such approaches comes from the need to define cryptographic protocols between the elements of the system in the first place, before security properties of the system be specified.

D2.3.3 The SysML profile

SysML (Systems Modelling Language) is a specification defined and promoted by the Object Management Group (OMG). OMG produces and maintains computer industry specifications for interoperable, portable and reusable enterprise applications in heterogeneous environments [38]. SysML is intended to provide simple but powerful constructs to model a wide range of system engineering problems. The goal of the SysML specification is to provide a “standard modelling language for a systems engineering to analyze, specify, design, and verify complex systems, intended to enhance system quality, improve the ability to exchange systems engineering information amongst tools and help bridge the semantic gap between systems, software and other engineering disciplines” [39]. As a legacy of the Unified Modelling Language (UML) specification, the SysML reuses a subset of that specification extending its semantics to emphasize requirements and parametric constraints [39]. Such features are particularly useful when security requirements are at stake.

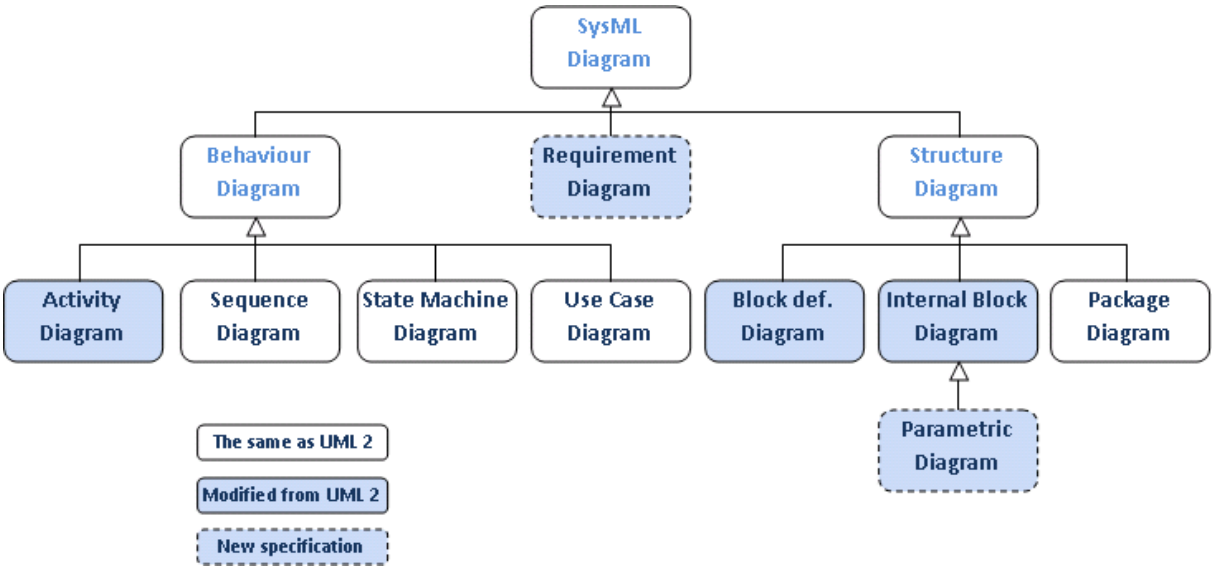
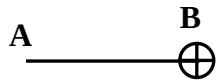
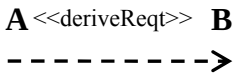
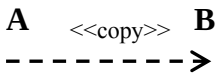
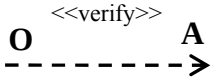


Figure 22 SysML Diagram Description

The SysML Requirement Diagram provides constructs to define and specify requirements that systems under design shall satisfy. Each requirement is specified in a single node with its name, its identifier and a brief plain text description of the requirement. The relationships between requirements can be established through links between nodes. Each path has a defined syntax. Based on this specification several operators have been implemented in TTool [13] to relate requirements in diagrams. Since these operators are used in Security Requirements Diagrams in the next subsections, they are listed and explained in Table 24. All links defined in SysML can be found in [40].

Table 24 Description of SysML symbols

Link	Symbol	Description
Composition		The requirement attached to the sharp edge **A** should be satisfied in order that the one attached to the crossed circle **B** be also fulfilled. The requirement **A** is part of the requirement **B**; therefore, the requirement **B** is fully described by the requirements it is composed of and a dependency relationship is established between **B** and its composing requirements.
Derive Requirement		The requirement attached to the sharp edge **A** should be satisfied in order that the one attached to the crossed circle **B** be also fulfilled. The requirement **A** is part of the requirement **B**; therefore, the requirement **B** is fully described by the requirements it is composed of and a dependency relationship is established between **B** and its composing requirements.
Copy		A copy operator between requirements **A** and **B** establishes a master/slave relationship between them in such way that the text of the slave **A** is a read-only copy of the text of the master **B**. For each slave a unique master should be related but several slaves are possible for each master.
Verify		A relationship between a requirement **A** and a test case **O** can determine whether the system that is represented in other diagrams, fulfils the requirement **A** or not. In our profile the test case **O** is expressed through nodes called *Tobservers*. Additional actions can be attached to observers if after verification the requirement **A** is not satisfied. Observers are defined with a name, a related diagram, an informal description of the property and an identifier that is used when the property is not satisfied. Observers are the medium through which system model and Requirement Diagrams are linked for verification purposes.

D.2.3.4 SysML for EVITA: Expected outcomes

In our proposed framework, security requirements are described with an approach close to the language of use case designers and system engineers. SysML is a standardized specification intended for software and hardware system engineering modelling. Models based on this profile satisfy interoperability and portability criteria. Moreover, SysML can be used within a Model Driven Approach. In EVITA, we have been using the domains defined by use cases to classify the security requirements determined in addition to their security properties and describing the relationships between security requirements and in particular their respective dependencies as explained in Table 24. This is the reason why security requirements are modelled in the SysML requirement diagram [39]. We are additionally working on the introduc-

tion of attack specifications into the parametric diagram in order to obtain a more complete description of requirements and their coverage.

- Security requirements are described in a way that relates to threats identification (i.e., trace security requirements aimed at threat mitigation or anti-goal prevention rather than security property or goal achievement). Security requirements also contain observers, which may be seen as test cases meant to be used for the formal verification (or simulation) phase. Observers may additionally be seen as a means to document requirements. This set of requirements and observers altogether provides a conceptual model of the security expectations of the system, abstracted from the literary description of use cases.
- Security requirements are defined using different relationships. We have more particularly used containment, dependency – deriveReqt, and reuse in different namespaces copy relationships.
- In our framework, threat modelling, security requirements, the Y-chart approach to hardware/software co-design [41], formal verification, and code generation, all are integrated into one environment (TTool) [13]. This feature is not present in environments presented in the related work section, for example in UMLSec [33]. TTool offers a unified environment for modelling and engineering embedded systems with security and real-time constraints.
- Requirement traceability is another important capability of our approach. Requirements can indeed be linked to attacks and to models of the system. SysML Requirements implemented in TTool can reference attacks of attacks trees, therefore facilitating the coverage study of attacks. We also intend, in a near future, to model attack trees with SysML parametric diagrams, therefore improving the integration of our solution. Parametric Diagrams are defined through graphical nodes that represent constraints or value types: constraints are meant to relate value types together. More precisely, value types could represent attacks while constraints are meant to model relations between attacks (or, and, etc.).
- Another advantage of using the SysML requirements diagram in EVITA is to show its relationship with other modelling elements to bridge the gap between traditional requirement management tools and other SysML models.

D.2.3.5 Environment Related Security Requirements

Prevent Malicious Modifications on the Environment Representation

This set of security requirements is organized under the more abstract requirement of preventing malicious modifications to the environment representation. This security requirement aims at preventing attackers from feeding wrong environmental information to the gateways and sensors. *Prevent Malicious Modifications on the environment representation* security requirement (GSR-1) has *Requirement Containment Relationship* which contains further security requirements such as *Integrity of Messages (GSR-1.1), Message Freshness(GSR-1.2), Authenticating Message Sources* (*GSR-1.3)* and *Environment Information Attestation (GSR-1.4).* More fine grained definition of these security requirements are explained in the next section. These requirements can be used alone or combined together as shown in Figure 23, in order to cut an attack tree node or different branches of attack tree(s).

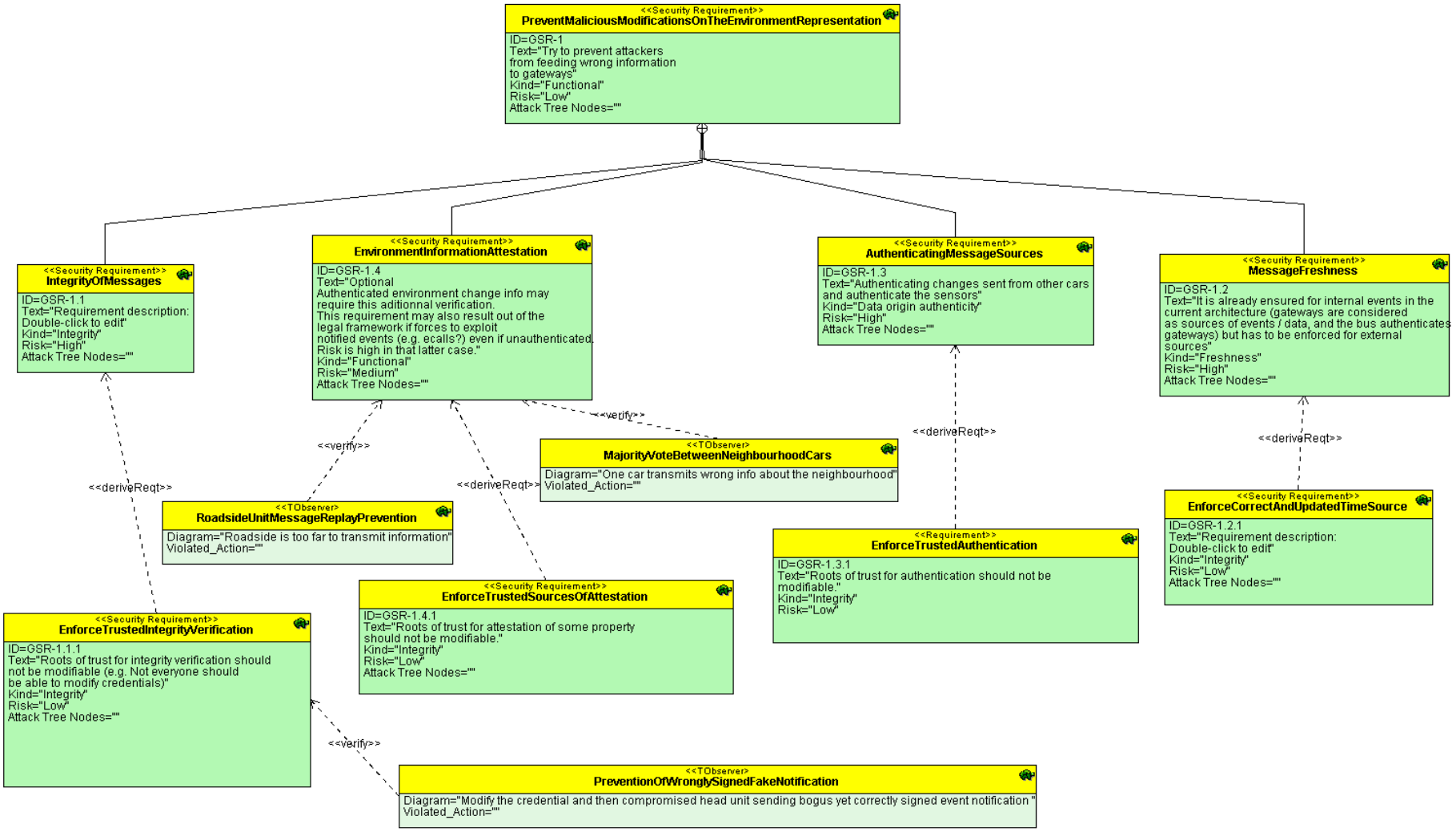


Figure 23 SysML Environment Related Security Requirements

D.2.3.6 Availability Requirements

This set of security requirements is focusing on properties that should be maintained despite denial of service attacks, coming either under the form of computational resource oriented DoS, network DoS, or even degradation of real-time constraints. ECU availability security requirement will ensure that: *"A service or a physical device providing a service is operational"*. *Ensure Availability of ECUs (ASR-1), requirement* has r*equirement containment relationship*, which contains *Ensure Bus Availability (ASR-1.1) Ensure CPU Availability (ASR-1.2), Ensure RAM Availability* (ASR-1.3) and *Ensure External Communication Device Availability (ASR-1.4)*. These requirements aim at preventing (temporary) denial of service attacks compromising the availability of their target at functional levels. An availability property applies to a service or to a physical device such as CPU, RAM or Bus. Furthermore *Availability of Highest Priority Functions (ASR-3)* and *Availability of Radio Medium (ASR-2)* must be ensured as shown in Figure 24.

D.2.3.7 Privacy Requirements

These security requirements are intended to protect the driver privacy in the cases when the relation between the vehicle and its use, and the identity of its owner or driver is confidential. The driver privacy requirement can be ensured having a *Controlled Access to Emergency Service Messages and Data* by the authorized entities, giving a *Privacy Policy for Disclosure of Information* about of the driver and his vehicle, and assuring *Unlinkability of Emergency Services on Critical Messages*. This *Unlinkability of Emergency Services* security requirement can be achieved by assuring *Car to External entities Message Privacy,* being *Unlikable Driver Identification between Services* and being *Unlinkable Time Ordering of Messages* as is shown in Figure 25.

D.2.3.8 Fake Command Requirements

This set of requirements is organized based on the abstract requirement to prevent sending fake commands. This security requirement will prevent attackers from sending wrong or fake commands from within or from outside the TOE. *Prevent Sending Fake Command (FSR-1)* requirement has *requirement containment relationship,* which contains further security requirements such as *Prevent Man In The Middle Attack (FSR-1.1), Prevent Replacement of Chips on Local Busses (FSR-1.2)* and *Protect ECU Flashing Process (FSR-1.3)* as shown in Figure 26.

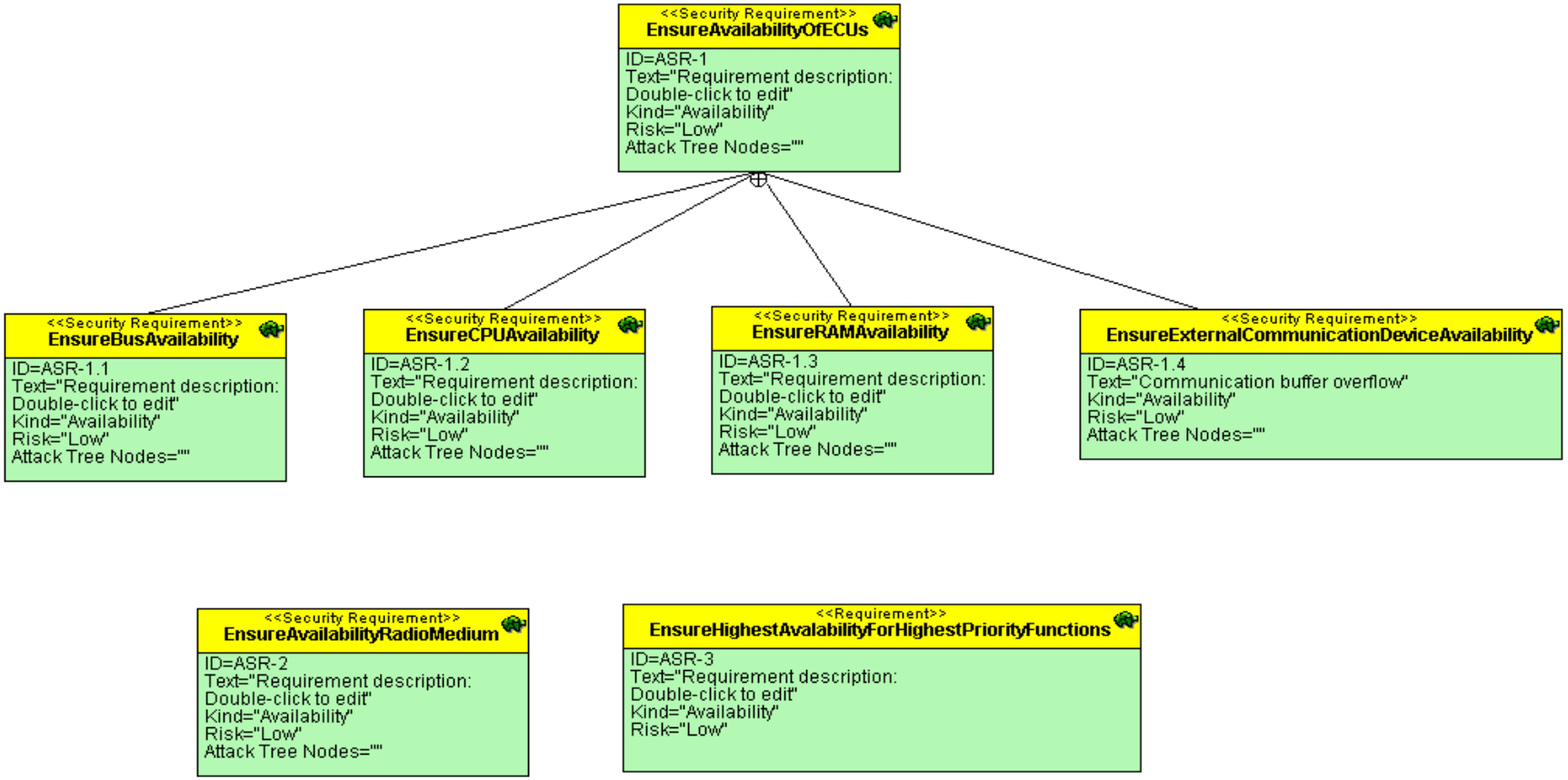


Figure 24 SysML Availability Security Requirements

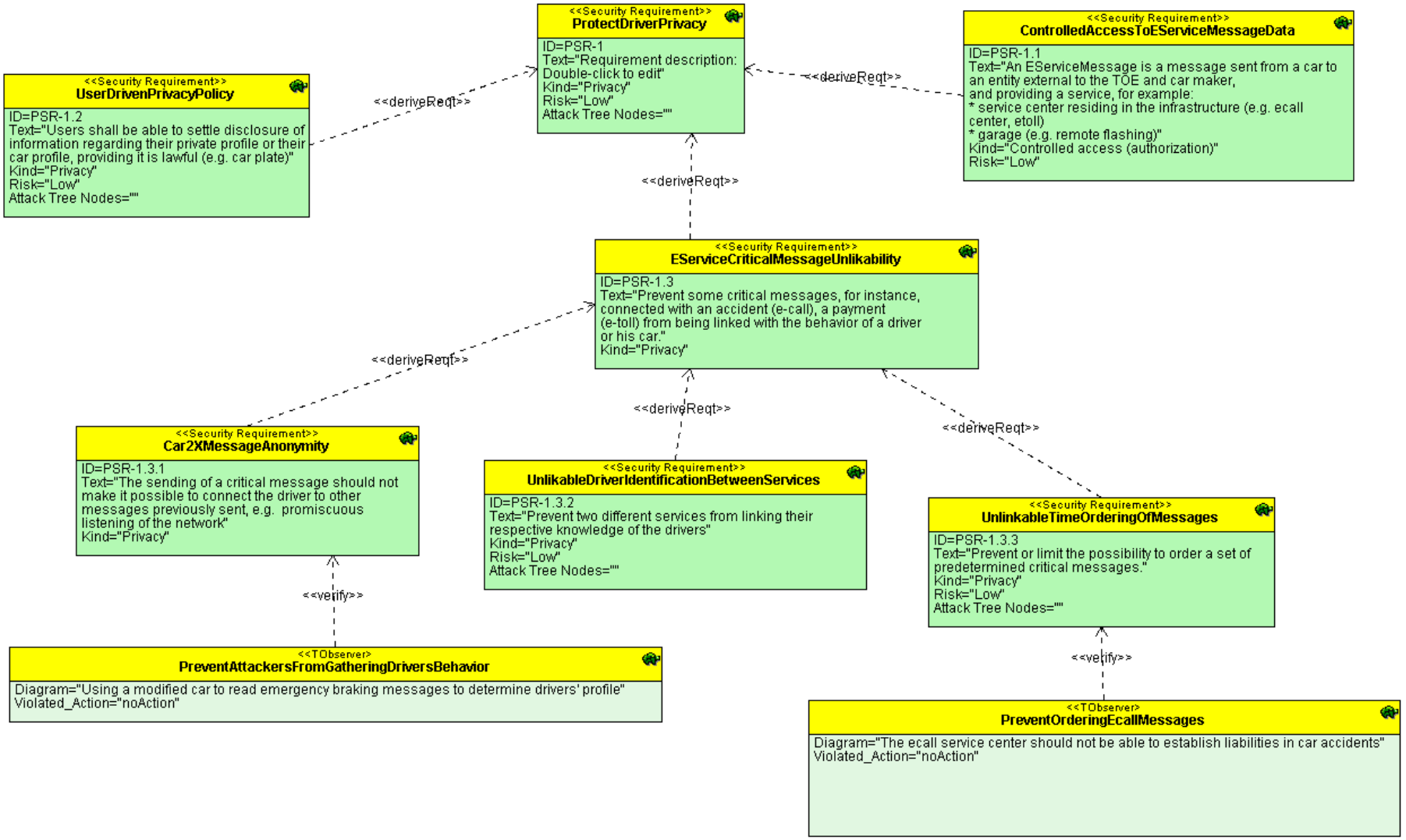


Figure 25 SysML Privacy Security Requirements

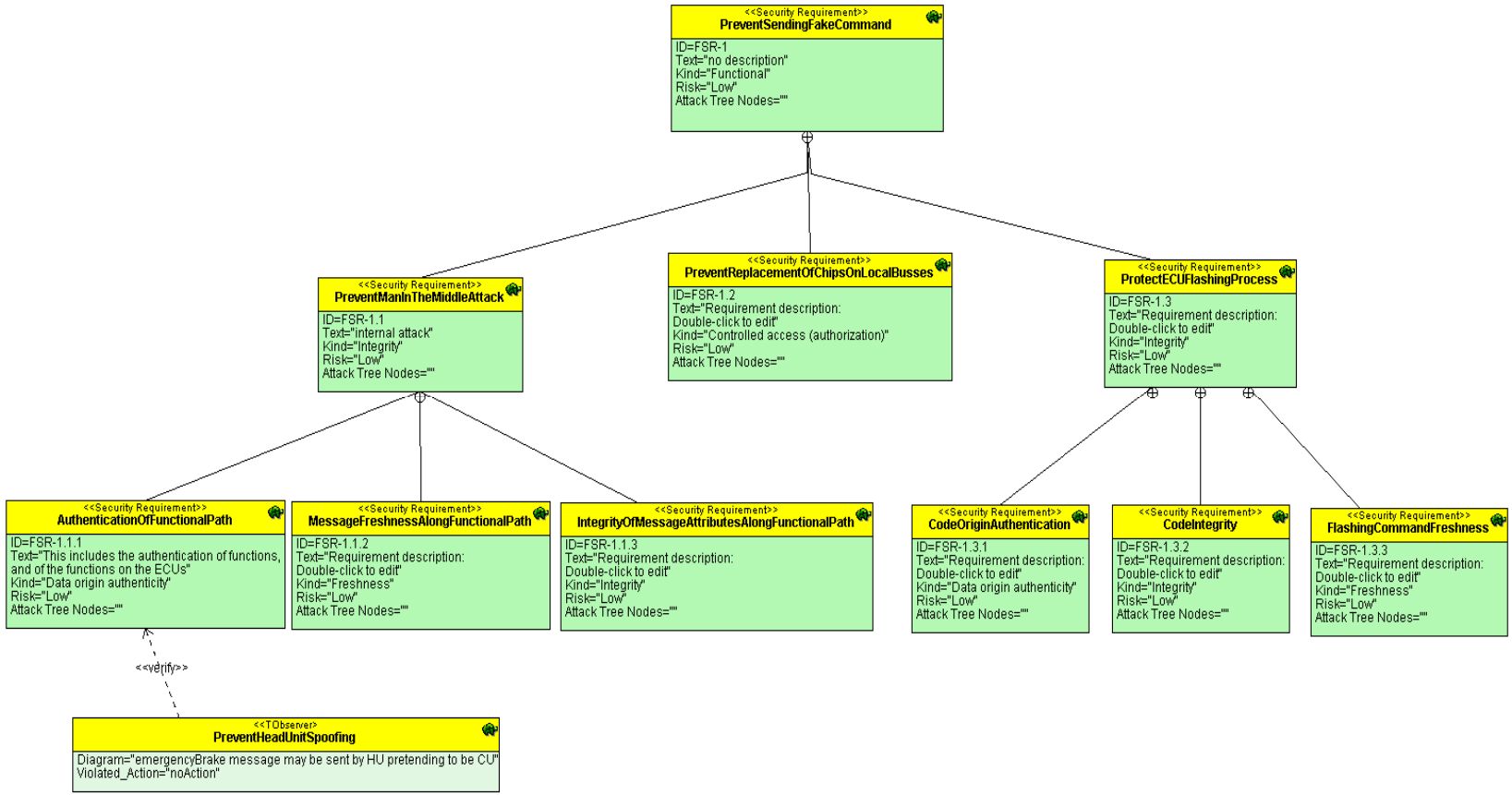


Figure 26 SysML Fake Command Security Requirements

D.2.3.9 Flashing Requirements

This set of security requirements will ensure that a flashing update takes place with authorized firmware, whose IPR is not endangered. Whenever a flashing process is performed, a *Controlled Access to Flash Memory* (FBSR-1) requirement may be specified. This will ensure that flash memory should be paired with their ECU to prevent flash replacement. Controlled Access to Flash Memory has a r*equirement containment relationship which* contains: *Controlled Access to Flashing Function (FBSR-1.1)*: This security property requires Integrity Property – *Code Integrity (FBSR-1.1.1) and Integrity of Firmware Update requirements (FBSR-1.1.1.1).* Whereas *Controlled Access Property – Controlled Access to Read from Flash (FBSR-1.2)*: This security property requires *Confidentiality Property – Confidentiality of Firmware Data (FBSR-1.2.1)* and *Confidentiality of Firmware Update (FBSR-1.2.1.1) requirements* as shown in Figure 26.

D.2.3.10 Braking DoS Requirements

This set of requirements is intended to ensure that braking manoeuvres that are triggered in the vehicle be done when they are required and by the authorized entities or actors in the manoeuvre. To ensure the availability of the services and entities required in braking situations is necessary *Prevent Brake Denial of Service attacks when Emergency Situations* happen and *Prevent broadcast (of this) Denial of Service attacks When Emergency Situations* happen too. To ensure that the breaking manoeuvres are triggered by the authorized entities is necessary to *Prevent Sending Fake Command* and *Authentication of Functional Path* to *Prevent Head Unit Spoofing*. A detailed view of these requirements can be found in Figure 28.

D.2.3.11 Security requirements coverage

To illustrate the application of the abovementioned security requirements, their coverage was analyzed for two attack trees (Automatic Brake Function and Unauthorized Braking) as shown in Table 25.

D.2.4 Functional and mapping views of use cases

Figure 29 shows the functional view of the “Safety Reaction: Active Brake” use case. Figure 30 shows the mapping view of the “Safety Reaction: Active Brake” use case. Figure 31 shows the functional view of the “Flashing per OBD” use case. Figure 32 shows the mapping view of the “Flashing per OBD” use case.

Messages between functions have been categorized into data or events. The system architecture is made of four CPUs. Each CPU comes with its own local bus on which is connected a bridge, a RAM, a flash memory, and possibly input/output devices. Four of these CPU blocks are considered: CU (Communication Unit), BU (Braking Unit), PTC (Powertrain Controller), CSC (Chassis and Safety Controller). The mapping of functions has been performed according to the use case description.

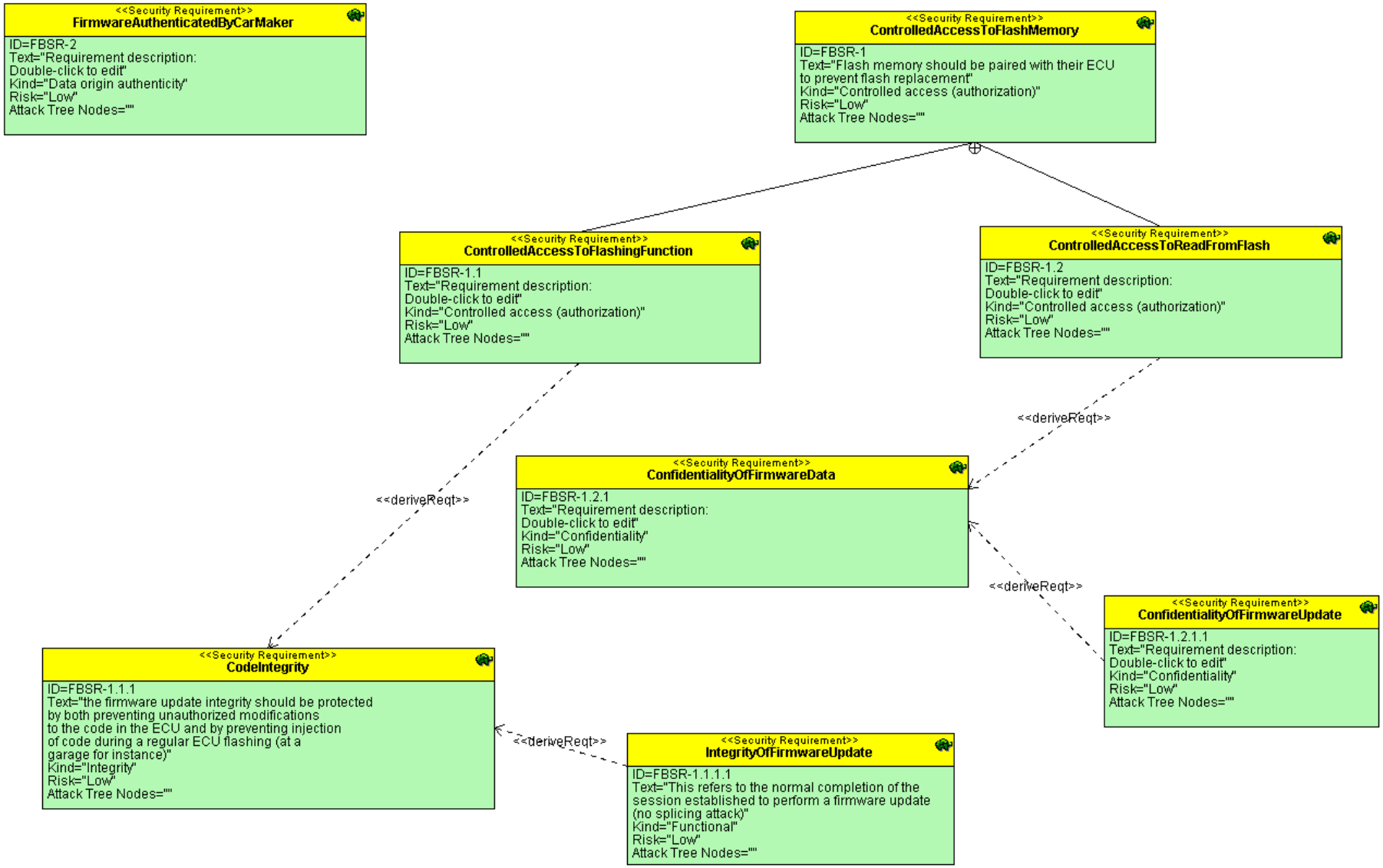


Figure 27 SysML Flashing Security Requirements

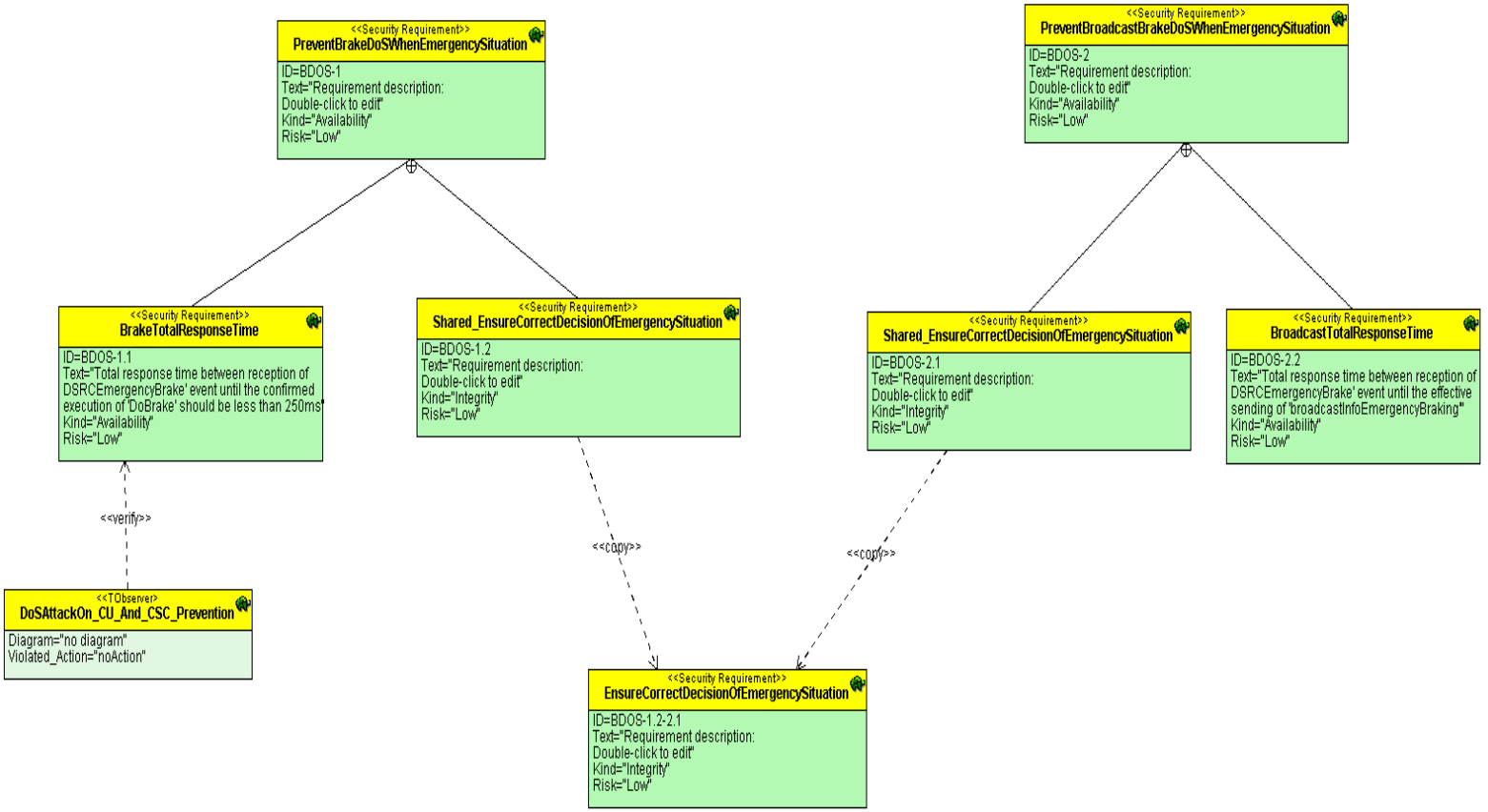


Figure 28 SysML Braking DoS Security Requirements

Table 25 Security Requirements Coverage – Attack Trees 8 and 9

Security Requirements				Attack Tree Nodes
Environment related Security Requirements – General Requirements (1)	Prevent Malicious Modifications of the Environment Representation	Integrity of Messages	Enforce Trusted Integrity Verification	8.2.1.2.1, 8.2.2.1.1.1, 9.2.1.3, 9.2.1.2, 9.2.1.1
		Environment Information Attestation	Enforce Trusted Sources of Attestation	8.2.1.2.1, 8.2.2.1.1.1, 8.2.2.1.1.2.1, 8.3.1, 9.2.1.1.1, 9.2.1.3
		Authenticating Message Sources	Enforce Trusted Authentication	8.2.1.2.1, 8.2.1.1.2, 8.2.1.1.1, 8.2.2.1.1.2, 8.2.2.1.1.2.1, 9.2.1.1.1, 9.2.1.3
		Message Freshness	Enforce Correct and Updated Time Source	8.2.1.2.1, 8.2.2.1.1.1, 8.2.2.1.1.2.1, 9.2.1.1.1, 9.2.1.3
Fake Command Requirements – General Requirements (4)	Prevent Sending Fake Command	Prevent Man In The Middle Attack	Authentication of Functional Path	8.1.1.1.1, 8.1.1.2.1, 8.2.1.3.1, 8.2.2.2
			Message Freshness along Functional Path	8.1.1.1.1, 8.1.1.2.1, 8.2.1.3.1, 8.2.2.2
			Integrity of Message Attributes along Functional Path	8.1.1.1.1, 8.1.1.2.1, 8.2.2.2
		Prevent Replacement of Chips on Local Busses		
		Protect ECU Flashing Process	Code Origin Authentication	8.1.2, 8.1.1.2, 8.2.1.3.1, 8.3.1
			Code Integrity	8.1.2, 8.1.1.2, 8.2.1.3.1, 8.3.1
			Flashing Command Freshness	8.1.2, 8.1.1.2, 8.2.1.3.1, 8.3.1
Privacy Requirements – General Requirements (3)	Protect Driver Privacy	Car2Car Message Anonymity		
		e-Call Message Privacy		
		Controlled Disclosure of Time		
Availability and Overhead Requirements – General Requirements (2)	Ensure Availability of ECUs	Ensure Bus Availability		9.1.2.2, 9.3.1.2, 9.1.2.3, 9.1.1.1, 9.1.1.2, 9.3.1.3, 9.3.2.1, 9.3.2.2, 9.3.3.1, 9.3.3.3
		Ensure CPU Availability		9.1.2.2, 9.3.1.2, 9.1.2.3, 9.1.1.1, 9.1.1.2, 9.3.1.3, 9.3.2.1, 9.3.2.2, 9.3.3.1, 9.3.3.3
		Ensure RAM Availability		9.1.2.2, 9.3.1.2, 9.1.2.3, 9.1.1.1, 9.1.1.2, 9.3.1.3, 9.3.2.1, 9.3.2.2, 9.3.3.1, 9.3.3.3
		Ensure External Communication Device Availability		9.1.2.1, 9.3.1.1, 9.3.3.3
	Ensure Availability Radio Medium			9.3.3.3
	Ensure Highest Availability for Highest Priority Functions			

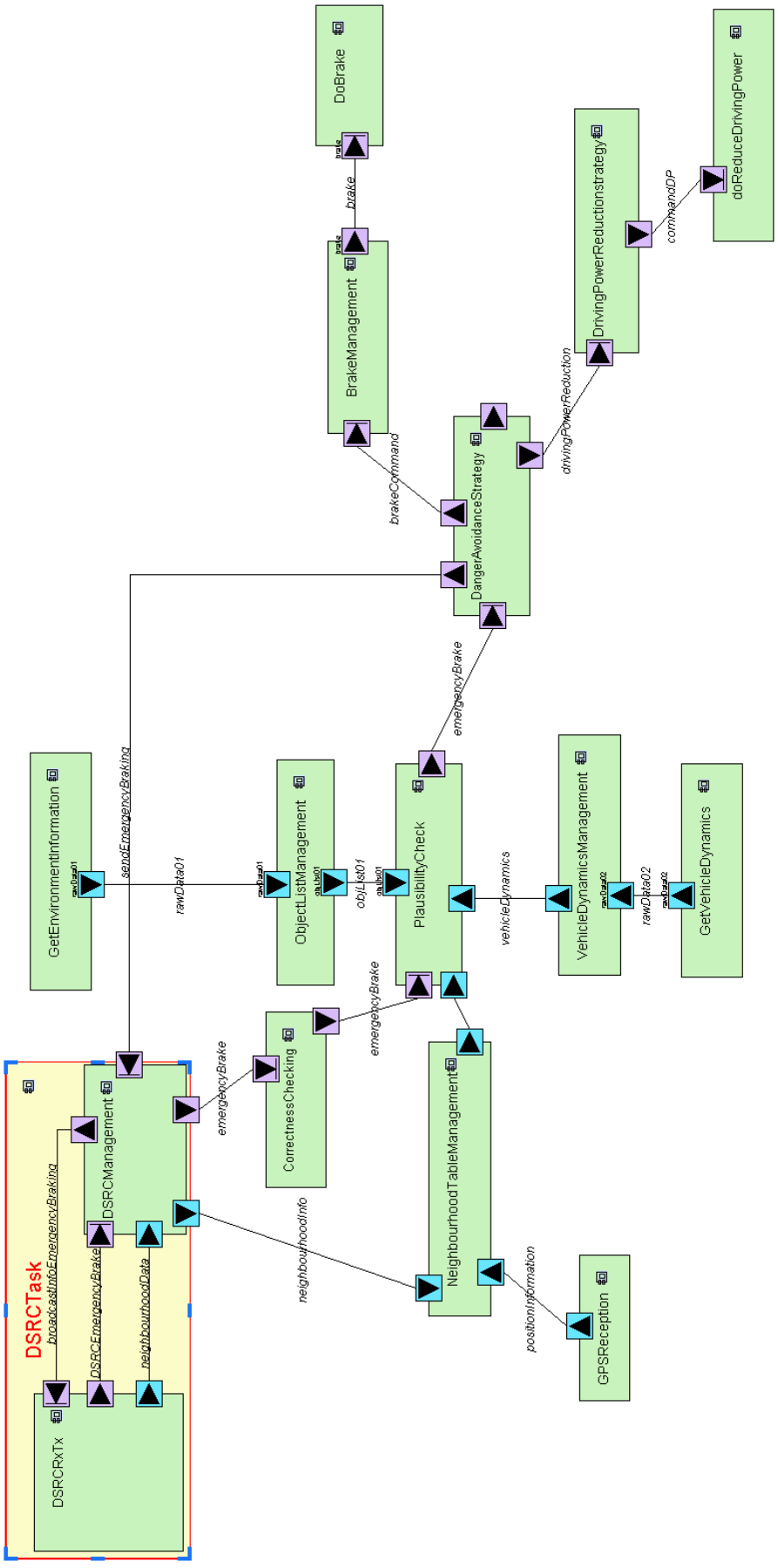


Figure 29 Functional view of Safety Reaction Active Brake

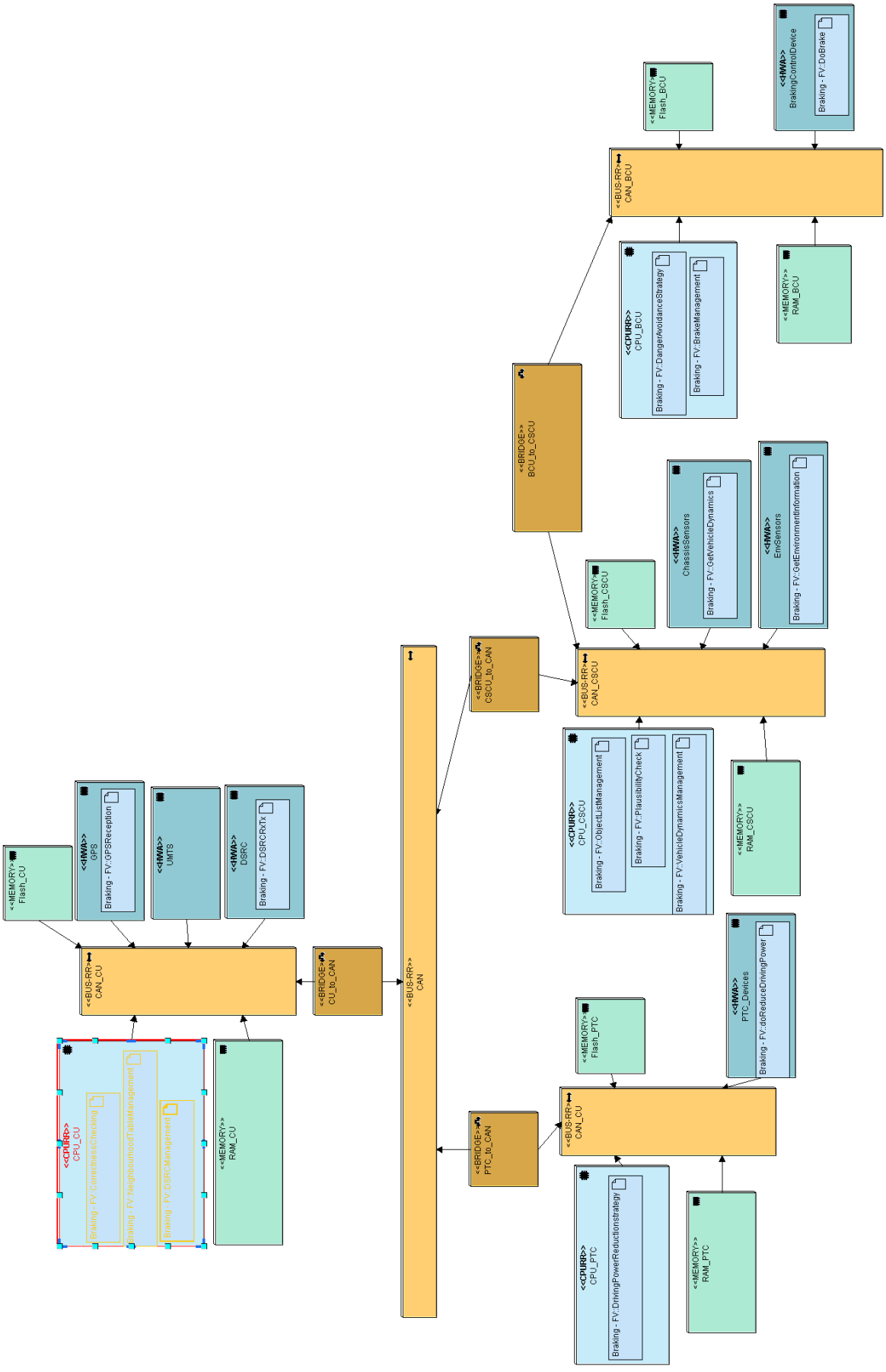


Figure 30 Mapping view of Safety Reaction Active Brake

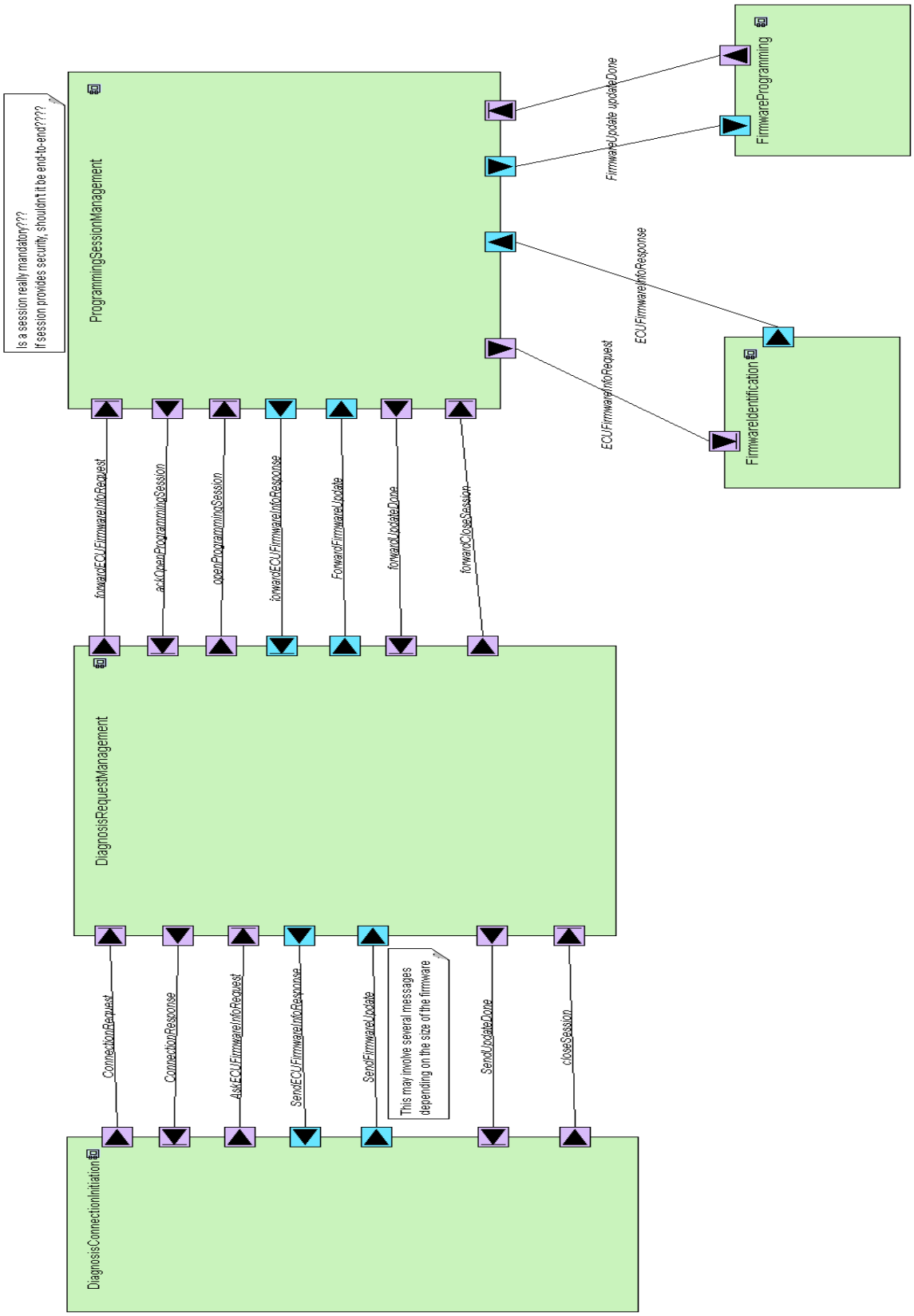


Figure 31 Functional view of Flashing per OBD

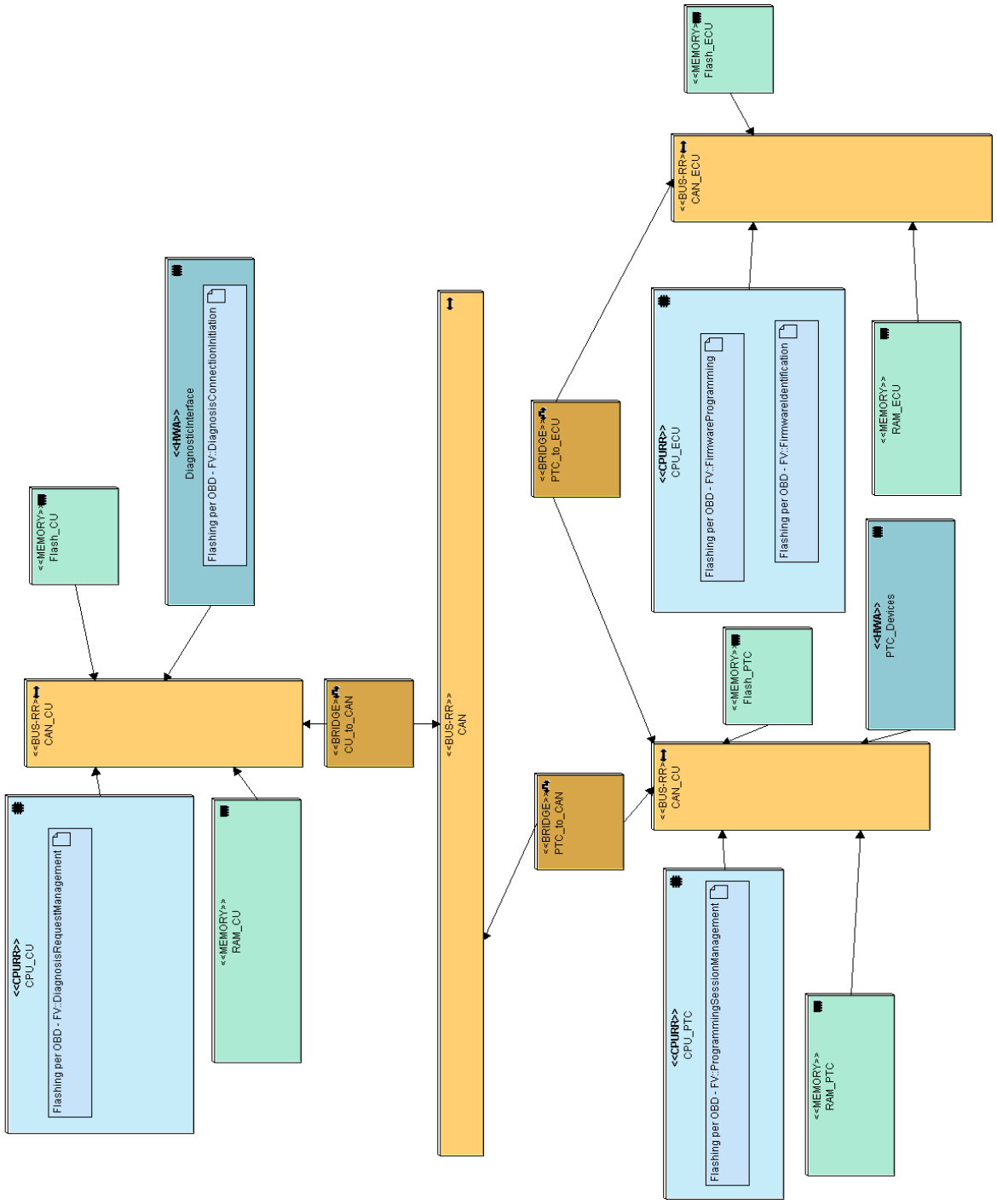


Figure 32 Mapping view of Flashing per OBD

References

[1] ISO/IEC 15408, "Information technology – Security techniques – Evaluation criteria for IT security", (3 parts).

[2] ISO/DIS 26262, "Road vehicles – Functional safety", ISO, draft, 2009 (10 parts).

[3] "MISRA Guidelines for safety analysis of vehicle based programmable systems", ISBN 978 0 9524156 5 7, MIRA, 2007.

[4] IEC 61508, "Functional safety of electrical/electronic/programmable electronic safety-related systems", IEC, 1998–2005 (8 parts).

[5] E. Kelling, M. Friedewald, T. Leimbach, M. Menzel, P. Saeger, H. Seudié, and B. Weyl, "Specification and evaluation of e-security relevant use cases", Deliverable D2.1 of EVITA, 2009.

[6] S. Gürgens, P. Ochsenschläger, and C. Rudolph, "Authenticity and Provability – a Formal Framework", Infrastructure Security Conference 2002, October 2002, Springer Verlag.

[7] S. Gürgens, P. Ochsenschläger, and C. Rudolph, "Parameter confidentiality", Informatik 2003 – Teiltagung Sicherheit, Gesellschaft für Informatik, 2003.

[8] S. Gürgens, P. Ochsenschläger, and C. Rudolph, "Abstractions preserving parameter confidentiality", Computer Security – ESORICS 2005, pp. 418–437, Springer Verlag, 2005.

[9] R. Grimm and P. Ochsenschläger, "Binding Cooperation. A Formal Model for Electronic Commerce", Computer Networks, Vol. 37, Issue 2, pp. 171–193, October 2001.

[10] P.H. Jesty and D.D. Ward, "Towards a unified approach to safety and security", Safety-critical Systems Symposium, Bristol, UK, February 2007.

[11] D.G. Firesmith, "Specifying Reusable Security Requirements", Journal of Object Technology (JOT), 3(1), Swiss Federal Institute of Technology (ETH), Zurich, Switzerland, pp. 61–75, January/February 2004.

[12] A. Aijaz, B. Bochow, F. Dötzer, A. Festag, M. Gerlach, R. Kroh, and T. Leinmüller, "Attacks on Inter Vehicle Communication Systems – an Analysis", 3rd International Workshop on Intelligent Transportation (WIT 2006) March 2006

[13] TTool: The TURTLE Toolkit. In http://labsoc.comelec.enst.fr/turtle/ttool.html.

[14] OMG Systems Modelling Language (OMG SysML™), Version 1.1

[15] D.D. Ward, P.H. Jesty and R.S. Rivett, "Decomposition scheme in automotive hazard analysis", SAE 2009 International Congress, Detroit, USA, April 2009, Paper 09AE-0248.

[16] B. Schneier, "Secrets and Lies – Digital Security in a Networked World", Wiley, New York, 2000, Chapter 21.

[17] ISO/IEC 18045, "Information technology – Security techniques – Methodology for IT security evaluation".

[18] Common Criteria Supporting Document – Mandatory Technical Document – Application of Attack Potential to Smartcards. Version 2.5, Revision 1, April 2008, CCDB-2008-04-001.

[19] "Abbreviated injury scale", Association of the Advancement of Automotive Medicine; Barrington, IL, USA (see www.carcrash.org).

[20] L. Apvrille, "TTool for DIPLODOCUS: an Environment for Design Space Exploration", Proceedings of the 8th international conference on New Technologies in Dsitributed Systems, Lyon, France, 2008.

[21] D. Knorreck, Ludovic Apvrille, Renaud Pacalet, "Fast Simulation Techniques for Design Space Exploration", 47 International Conference TOOLS EUROPE, Zurich, Switzerland, 2009.

[22] L. Yin, J. Liu and X. Li, "Validating Requirements Model of a B2B System", 8th International Conference on Computer and Information Science, IEEE/ACIS, 2009.

[23] P. Giorgini, F. Massacci, J. Mylopoulos and N. Zannone, "ST-Tool: A CASE Tool for Security Requirements Engineering", 13th International Conference on Requirements Engineering, IEEE, 2005.

[24] S. Graf, I. Ober and I. Ober, "A Real-Time Profile for UML", VERIMAG, France, in http://www-verimag.imag.fr/.

[25] The Unified Modelling Language (UML) in http://www.uml.org/

[26] W. Sun, J. Wu, Y. Xiong, "Methodological Support for Service-oriented Design with rCOS", International Symposium on Information Engineering and Electronic Commerce, IEEE, 2009.

[27] O. Kath, M. Soden, M. Born, T. Ritter, A. Blazarenas, M. Funabashi, C. Hirai, "An Open Modelling Infrastructure integrating EDOC and CCM", 7th International Enterprise Distributed Object Computing Conference, IEEE, 2003.

[28] W. Heaven and A. Finkelstein, "A UML Profile to Support Requirements Engineering with KAOS", IEE Proceedings, 2004.

[29] A. Moore, "Extending the RT Profile to Support the OSEK Infrastructure", 5th International Symposium on Object-Oriented Real-Time Distributed Computing, IEEE, 2002.

[30] The Object Management Group (OMG), "The EDOC specification" in http://www.omg.org/mda.

[31] Z. Wang, X. Yu, G. Pu, L. Feng, H. Zhu and J. He, "Execution Semantics for rCOS", 15th Asia-Pacific Software Engineering Conference, IEEE, 2008.

[32] P. Bertrand, R. Darimont, E. Delor, P. Massonet, A. van Lamsweerde, "GRAIL/KAOS: An Environment for Goal-Driven Requirements Engineering", CEDITI-UCL and Université Catholique de Lauvain, Belgium.

[33] J. Jürjens, "UMLsec: Extending UML for Secure Systems Develpment", 5th International Conference on the Unified Modeling Language, pp. 412-425, 2002.

[34] J. Jürjens and P. Shabalin, "Automated Verification of UMLsec Models for Security Requirements", Munich University of Technology, Germany, in http://www4.in.tum.de/~juerjens, http://www4.in.tum.de/~shabalin.

[35] K.P. Peralta, A.M. Orozco, A.F. Zorzo, F.M. Oliveira, "Specifying Security Aspects in UML Models", Pontifical Catholic University of Rio Grande do Sul, Brazil.

[36] T. Lodderstedt, D.A. Basin, J. Doser, "SecureUML: A UML-based Modeling Language for Model Driven Security", 5th International Conference on the Unified Modeling Language, pp. 426-441, 2002.

[37] M. Boreale and M.G. Buscemi, "Experimenting with STA, a Tool for Automatic Analysis of Security Protocols", Symposium on Applied Computing, ACM, Lecture Notes in Computer Science, pp. 281-285, 2002.

[38] The Object Management Group (OMG) in http://www.omg.org.

[39] The OMG Systems Modelling Language in http://www.omgsysml.org

[40] OMG, "OMG Systems Modelling Language (OMG SysML) specification", Final adopted specification ptc/2006-05-04, 2007.

[41] P. Lieverse, P. van der Wolf, E. Deprettere, K. Vissers, "A Methodology for Architecture Exploration of Heterogeneous Signal Processing Systems", Workshop on Signal Processing Systems, IEEE, 1999.