HoliSec - HOLIstic Approach to Improve Data SECurity

Dnr 2015-06894

FORDONSSTRATEGISK FORSKNING OCH INNOVATION

Document Title	A State-of-the-Art Report on Vehicular Security
Document Type	Deliverable
Document Number	D1.2
Document Responsible	Thomas Rosenstatter, thomas.rosenstatter@chalmers.se, Chalmers
Document Version	1.0
Document Status	Final
Dissemination Level	Public
Last Change	19th April 2017

Project Acronym	HoliSec
Project Title	HOLIstic Approach to Improve Data SECurity
Research Program	Vinnova/FFI (Fordonsutveckling/Vehicle Development), Sweden
Diary Number	2015-06894
Project Duration	2016 - 2019
Project Coordinator	Atul Yadav, atul.yadav@volvo.com, Volvo AB

This page is intentionally left blank

Executive Summary

This deliverable (D1.2 A State-of-the-Art Report on Vehicular Security) presents the results and achievements of sub-work package WP1.2 (State of the art) of the HoliSec project. The goal of this deliverable is to provide insight into the state of current research frontiers in vehicular security. It also serves as an introduction to the issues related to vehicular security with an extensive reference list for more detailed studies. Finally, it can be used as background on which to base further investigations. The FFI HEAling Vulnerabilities to ENhance Software Security and Safety (HEAVENS) project produced the deliverable "D1.2 State of the art", HoliSec uses this deliverable as a baseline and reflects the current state of the art.

The report includes discussions about hardware and operating system level security, privacy and authentication issues, internal and external communication security, secure software development processes, security evaluations and certifications, standardization organizations and related research projects. Many topics are discussed very briefly, since an exhaustive treatment would certainly be out of the scope of this deliverable.

This page is intentionally left blank

Contributors

Editor(s)	Affiliation	Email
Aljoscha Lautenbach	Chalmers	aljoscha@chalmers.se
Tomas Olovsson	Chalmers	tomas.olovsson@chalmers.se
Thomas Rosenstatter	Chalmers	thomas.rosenstatter@chalmers.se
Contributor(s)	**Affiliation**	**Email**
Mafijul Islam	ATR, GTT, Volvo AB	mafijul.islam@volvo.com
Pierre Kleberger	Chalmers	pierre.kleberger@chalmers.se
Aljoscha Lautenbach	Chalmers	aljoscha@chalmers.se
Boel Nelson	Chalmers	boeln@chalmers.se
Nasser Nowdehi	Volvo Car Corporation	nasser.nowdehi@volvocars.com
Tomas Olovsson	Chalmers	tomas.olovsson@chalmers.se
Thomas Rosenstatter	Chalmers	thomas.rosenstatter@chalmers.se
Riccardo Scandariato	Chalmers	riccardo.scandariato@chalmers.se
Katja Tuma	Chalmers	katjat@chalmers.se

HoliSec Consortium

Document Change History

Version	Date	Contributor	Description
0.5	2016-12-20	Thomas Rosenstatter	Submitted for consortium feedback.
1.0	2017-04-19	Thomas Rosenstatter	Final version, incorporated consortium feedback.

This page is intentionally left blank

Contents

List of Figures

List of Tables

Acronyms

AMP Arbitration on Message Priority.

AP Access Point.

ASIL Automotive Safety-integrity Level.

ASLR Address Space Layout Randomization.

AUTOSAR Automotive Open System Architecture.

AVB Audio Video Bridging.

BSS Basic Service Set.

C2C-CC Car 2 Car Communication Consortium.

CA Certificate Authority.

CAM Cooperative Awareness Message.

CAN Controller Area Network.

CAN-FD Controller Area Network flexible data-rate.

CARONTE Creating an Agenda for Research on Transportation sEcurity.

CD Collision Detection.

CEN European Committee for Standardization.

CERT Computer Emergency Response Team.

CPS Cyber Physical System.

CPU Central Processing Unit.

CRC Cyclic Redundancy Check.

CRL Certificate Revocation List.

CSMA Carrier Sense Multiple Access.

DENM Decentralized Environmental Notification Message.

DoIP Diagnostics over IP.

DoS Denial of Service.

DSRC Dedicated Short Range Communication.

E/E Electrical and/or Electronic.

ECC Elliptic Curve Cryptography.

ECDSA Elliptic Curve Digital Signature Algorithm.

ECU Electronic Control Unit.

ePTI electronic Periodical Technical Inspection.

ETSI European Telecommunications Standards Institute.

EUCAR European Council for Automotive R&D.

GPS Global Positioning System.

HEAVENS HEAling Vulnerabilities to ENhance Software Security and Safety.

HoliSec HOLIstic Approach to Improve Data SECurity.

HSM Hardware Security Module.

IDS Intrusion Detection System.

IEC International Electrotechnical Commission.

IEEE Institute of Electrical and Electronics Engineers.

IPS Intrusion Prevention System.

ITS Intelligent Transportation System.

LIN Local Interconnect Network.

MAC Message Authentication Code.

MOST Media Oriented Systems Transport.

NDM Network Device Monitor.

NFC Near Field Communication.

NHTSA National Highway Traffic Safety Administration.

OBD-II On-Board Diagnostics II.

OS Operating System.

RDS Radio Data System.

RFID Radio-frequency Identification.

RSU Road-side Unit.

SDL Security Development Lifecycle.

SoC System on a Chip.

TDMA Time Division Multiple Access.

TTE Time-Triggered Ethernet.

UDS Unified Diagnostic Services Protocol.

V2I Vehicle-to-Infrastructure.

V2M Vehicle-to-Mobile.

V2V Vehicle-to-Vehicle.

V2X Vehicle-to-X.

VANET Vehicle ad-hoc Network.

WLAN wireless LAN.

WSN Wireless Sensor Network.

Chapter 1

Introduction

The internet has now reached our vehicles and many new services will be introduced in the coming years. Some services target drivers and passengers such as navigation and driver assistance systems, and other focus on the vehicle itself such as remote diagnostics and remote software updates. Most vehicle manufacturers have plans to offer a fairly large number of services and we now face the challenge to implement new functionality without sacrificing traffic safety. The vehicle is a complex safety-critical system with components that must function at all times, and security problems should never result in safety problems or in an immediate halt of all systems. Instead the vehicle must operate in a degraded and fail-safe mode when under attack and when security problems have been detected.

Today's vehicles have an internal network consisting of 30 to 100 computers or Electronic Control Units (ECUs). The internal network is of the size of a small company and internal security is currently largely missing. The software in a modern vehicle contains tens of millions of lines of code with a total size of more than 100 MBytes [1]. This vehicle will now be connected to the infrastructure around it, i.e. to Road-side Units (RSUs), to other vehicles and to the internet. Therefore, it is imperative that security is properly addressed before these new services are introduced.

1.1 The connected Vehicle

Communication between vehicles and the outside world will in almost all cases be wireless. Exceptions may be found in repair shops and when vehicles are parked. It is possible to access the internal network by connecting a device directly to the internal busses of a vehicle, for example in a repair shop to diagnose problems and to update the software, but also by vehicle owners and other people in order to "enhance" or change the vehicle's functionality. With a sound security design, it should not matter whether the communication is wired or wireless. However, physical modification of ECUs and the possibility to physically attach devices to the internal network must be paid special

attention to, as it can not be ruled out that the vehicle owner modifies or adds equipment to the vehicle that interferes with its normal functionality.

Vehicular Communication is divided into two or sometimes three categories, collectively called Vehicle-to-X (V2X):

- *Vehicle-to-Infrastructure (V2I)* communication: New services will be implemented that communicate with roadside equipment (i.e. new technical infrastructure). Most of those serices are related to safety, for example to alert drivers about traffic lights, speed limits or to inform about road works ahead.
- *Vehicle-to-Vehicle (V2V)* communication: This is the area most researchers and application developers focus on. Typical services are anti-collision systems such as early break warnings from other vehicles, information about emergency vehicles approaching, synchronized lane change support, traffic jam ahead warnings, and services facilitating the driving experience such as vehicle platooning.
- *Vehicle-to-Mobile (V2M)* communication: Communication with a mobile device, for example via Bluetooth or Near Field Communication (NFC). In the rest of this document, we will not include V2M communication in the V2X concept unless explicitly stated.

Different communication technologies are used for vehicular communication: Wireless LAN (WLAN) (IEEE 802.11a,b,g or n) can be used to connect vehicles to conventional access points, for example to download multimedia contents when parked at home. Mobile phones offering Bluetooth connectivity for hands-free operation is already implemented, often without considering that telephones also have a GPRS/3G data connection to the internet and therefore may bridge the vehicle with the internet. A new standard for Dedicated Short Range Communication (DSRC) has been developed. It is based on the new IEEE 802.11p standard which is intended to be used for communication with the infrastructure around the vehicle and with other vehicles. There are also many other devices communicating with the vehicle, for example wireless keys, Radio-frequency Identification (RFID) cards identifying drivers, radio communication for traffic information (RDS) and navigation (GPS).

Communication patterns and the actual technology used depend highly on the application, some are classical client-server applications where the vehicle connects to a server or a portal for provisioning of a specific service. Other services communicate with RSUs or are based on ad-hoc communication between different parties where short-lived Vehicle ad-hoc Networks (VANETs) are formed.

For some of the services it is essential that the parties are fully authenticated and their identities are known by all participants, and for other services privacy can be more important than being able to identify each other. A compromise may sometimes be acceptable, for example where vehicles can be anonymous to each other but RSUs are able to do proper authentication. Protection against non-repudiation and Sybil attacks

(multiple identities) is important for some services. Even if the parties do not want to reveal their identities to each other, there should be some limitations to what damage they may cause, and it should always be possible for an authorized party or authority to reveal their identities.

1.2 New Applications and Services

There is a high demand for applications and services around the connected vehicle, and vehicle manufacturers have long lists of applications they would like to offer to their customers. There will likely be something similar to an "AppStore" in the vehicle where the owner, driver and passengers can choose to install both free applications and subscribe to different services. In addition, mobile phones and other hand-held devices will be seamlessly integrated into the vehicles, enhancing the driving experience.

Applications will be offered by many parties such as the vehicle manufacturer, government organizations, trusted third parties (cooperating with the manufacturers), and independent third-party application developers. Some applications may be mandatory and offer services from legal authorities, others are safety improving services (e.g. driver assistance/autopilot and accident reporting systems) and yet other are services the owner, driver and passengers would like to install.

It yet remains to define an architecture that allows third party applications to run in the vehicle environment in a safe and secure way. Many solutions such as certification, sandboxing and isolation have been proposed, but it will take many years before vehicle manufacturers can allow third parties to freely develop software for the vehicles and still be able to guarantee the safety of the vehicle.

The services can roughly be categorized as:

- *Services improving the driving experience and safety*: They give advice and notify drivers about events. Examples include vehicle platooning (cooperative driving) and collision avoidance systems that give early warnings and notifications to drivers. Vehicles talk to each other and are fully aware of other vehicles' plans and act accordingly, for example by notifying drivers to give way to emergency vehicles approaching at high speed.

- *Critical safety services*: For example emergency actions from the vehicle when a crash is impossible to avoid and to call for help when sensors detect that an accident has occurred [2]. Many of these services are active and the vehicle will take action and help the driver in difficult situations.

- *Traffic optimization*: Information is spread among vehicles about road conditions, congestion problems, accidents and overall traffic throughput.

- *Commercial services*: For instance automatic payments of parking, road tolls and taxes based on when and where a vehicle has been driven.

- *Subscriptions to improved vehicle functionality*: The vehicle may have more functionality than the owner has initially purchased; it may be possible to use the internet to purchase or for a limited time rent functionality such as *automatic parking support* without having to visit the vehicle dealership. Vehicles may also be remotely diagnosed and the vehicle manufacturers can offer updated services and patch software in the field, not only in the repair shops.
- *Services unique to the driver, not to the vehicle*: Insurances may in the future follow the driver and not necessarily be tied to a particular vehicle, as is already the case in the U.K. Drivers may subscribe to services that are available to them regardless of what vehicle they are driving.

The proposal of the U.S. Department of Transportation (DoT) highlights that national departments also see this promising technology (V2V communication) as an enhancement for traffic safety. On the 13^{th} December 2016 the DoT proposed a rule that all light-duty vehicles are required to have an active V2V and V2I communication module in order to increase traffic safety and prevent accidents due to the exchange of information between the vehicles and infrastructure [3].

1.3 Summary

The use of many different communication technologies, different types of communication requirements, real-time requirements, authentication services mixed with anonymity, etc., makes security work very challenging. In addition we will soon have a large number of applications in our vehicles and several of them will be third party applications.

It is obvious that all communicating systems must be protected against external threats (attackers), but there are also many different parties involved with the vehicle that do not necessarily trust each other. The driver may not always have the same interests as the owner of the vehicle, for example with respect to sharing information about where and how aggressively the vehicle has been driven. Furthermore, owners and drivers may not always be trusted by the vehicle manufacturers, since there may be optional services offered by the manufacturers that the owner must purchase or subscribe to, to be able to use. If there is an easy way to obtain such services for free, e.g. to patch the software in the vehicle, many owners will probably use this opportunity.

The remaining structure of the report is as follows. Chapter 2 presents the complexity of modern/future vehicles and the challenges in securing future vehicles, which are discussed in more detail in later chapters. Chapter 3 highlights the necessity of securing the vehicle. Several cases are shown where the vehicle could be hacked from a wide range (the internet), or a closer range. Privacy models that are applicable for the automotive sector are discussed in Chapter 4. With the increased communication to the internet or with other vehicles, new threats are likely to appear and also threats known from other areas will emerge. Threats known from other areas are listed and discussed in Chapter 5. Chapter 6 addresses the communication technologies used for external communication, e.g. with other vehicles, infrastructure, or internet services. Furthermore, this Chapter

discusses methods that are needed when communicating with other parties. Securing the internal network is necessary to identify falsified messages and ensure the safety of the passengers. Chapter 7 discusses these issues. Security mechanisms provided by microcontroller or known from operating systems are shown in Chapter 8. Developing a secure software starts in the design phase, threats need to be identified in early stages in order to create a holistic software architecture that considers security right from the beginning. Chapter 9 describes different techniques for threat modeling. The activities during the design stage for developing secure software solutions can be taken from Chapter 10. Chapter 11 presents current best practices and methodologies with regards to the secure software development lifecycle. Chapter 12 is about the certification and evaluation of security functionality. Chapters 13 and 14 present standards and research projects related to the issues raised in this report. Finally, Chapter 15 closes with some final remarks.

Chapter 2

Complexity and Security Challenges

This chapter contains a discussion of the specific challenges we face when working with security in the vehicular domain.

Security researchers and analysts often face the problem to reach the market in time, not only in the vehicular domain. This often leads to service implementations which solve security problems one at a time, while waiting for standards and established methods to emerge. For obvious reasons this approach will not be successful in the long run.

More general methods that can be applied to a large number of applications and use cases must be considered. One such example is the framework developed by the European EVITA project [4]. Standardization work to specify security frameworks for V2X services is also in progress. The European Telecommunications Standards Institute (ETSI) *Intelligent Transportation System (ITS) station* is an attempt to standardize V2X communication nodes [5] and the IEEE standardized protocols for wireless communications (IEEE 1609 and IEEE 802.11p). Standardization and certification are discussed in more detail in chapter 13.

Since vehicle manufacturers act on multinational markets where they have to comply with different legal requirements, there may also be conflicting requirements, e.g. the trade-off between privacy and traceability. General security solutions are needed which provide the possibility to adapt the functionality of a vehicle model depending on the target country.

Further security complications arise from the vast number of services, communication technologies and protocols that must be supported. There are also real-time requirements and safety aspects that affect security and the functionality of the systems. In the following, we will show the complexity of the problem and highlight the challenges we face in each area.

2.1 Trust and Privacy Problems

Many parties are involved during the lifetime of a vehicle, which leads to several trust and privacy issues. It is obvious that people with no legitimate access to a vehicle (i.e traditional attackers) must be considered in a threat model. However, serious security problems may also be caused by authorized people with access to the vehicle.

Trust is generally problematic in the vehicular environment. For example, the owner of a vehicle is not necessarily trusted by the vehicle manufacturer, the driver may not be fully trusted by the owner, and repair shops may not be fully authorized or trusted by the vehicle manufacturer or the vehicle owner (i.e. they should not have full access to all data and programs in the vehicle). There are also third party program developers who want to offer services which none of these parties can fully trust. All involved parties have different notions of what is important to address, complicating the security work even further.

The complexity of security problems can be seen in Figure 2.1 in which different parties, communication technologies, and security attributes that must be considered are listed [6]. A security assessment of the vehicle must consider all possible combinations of actors, communication technologies, paths and security attributes, a quite complex task. This is discussed in more detail in Section 12.1.

Many parties such as the owner have physical access to the vehicle and its internal network. It is not unlikely to assume that if it is easy to enhance the vehicle's functionality, this will become popular. The internal vehicle architecture must therefore have a certain degree of protection against "attackers" and insiders with physical access to the vehicles.

Privacy issues are important. Most vehicle owners and drivers do not want to reveal their identities to everyone. However, being completely anonymous opens up for attacks against many services. Sybil attacks, i.e. to be able to use multiple identities could be useful for example by an attacker to spread false information about congestion in order to have the road for him/herself. To mitigate this problem, the use of pseudonyms has been suggested. When needed, for example for legal reasons, a trusted party will be able to reveal the real identity behind a pseudonym.

Privacy issues related to different applications must also be addressed. The owner may want to track the vehicle and how it is driven, something the driver may not always want. Some external services such as automatic payment of road tolls may also need to access private information. These and other privacy problems are further discussed in Chapter 4.

2.2 External Communication

Communication with the outside world can be done using many different technologies. Examples include DSRC (e.g. IEEE 802.11p), normal WLAN communication (802.11a,b,g,n), and cellular communication (GSM, GPRS, 3G, 4G). These technologies are often used by traditional client-server applications which possibly connect to the internet, Bluetooth

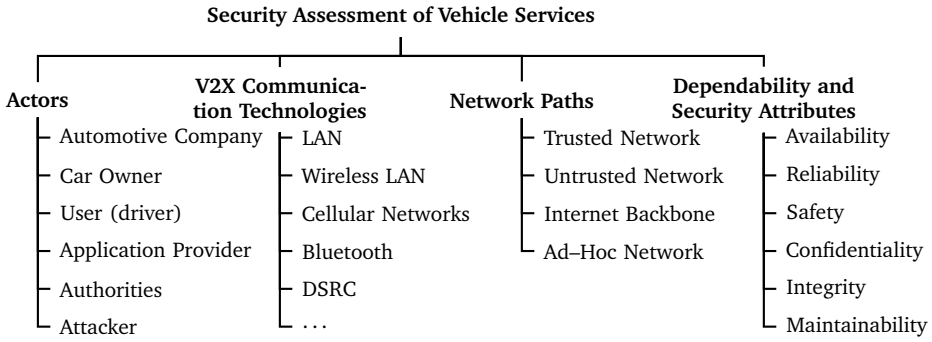

Figure 2.1: Security Assessment Tree

devices, or NFC devices. The large number of technologies complicates security work since several communication stacks necessarily lead to a bigger attack surface. It may also be hard to know which traffic should be allowed on what interface.

Through gateway ECUs which connect several busses, external devices are also able to send messages to the internal network, for that reason gateway ECUs need to be especially well secured against intrusions.

Chapter 6 provides a more detailed description. The different communication technologies and related protocols for external communication are briefly described in Section 6.1, and a discussion about their security properties can be found in Section 6.4.

2.3 Hardware and OS Security

Due to cost and power constraints, the ECUs in a vehicle are limited in terms of processing power and memory capacity, and all security measures must take this into account. Usually cheap "off-the-shelf" hardware is preferred for this reason, which may not have hardware support for cryptographic functions. As security is becoming even more important in the embedded domain, units with cryptographic hardware support will probably be deployed increasingly.

Modern Central Processing Units (CPUs) have the ability to distinguish between executable and non-executable memory spaces, which makes certain kinds of attacks harder to perform (e.g. the traditional buffer overflow attack).

Another CPU feature that is generally seen to be beneficial for security is hardware virtualization support. Virtualization can be used to create process instances which are isolated from one another, and which also have no direct access to real hardware resources. This is also an upcoming trend in embedded systems, but it is currently not used.

Physically securing ECUs against tampering is another challenge. This is called tamper resistance and is further discussed in Chapter 8. An attacker might for instance try to directly read the memory of an ECU to retrieve encryption keys. One approach could

be to erase the keys permanently if the physical case is opened, which presents both technical and practical problems.

Also related to physical attributes are side channel attacks, which aim to break a cryptosystem based on information related to its physical implementation, e.g. timing information or power consumption.

Much ECU software currently runs directly on the microprocessors, without an Operating System (OS). This means that security mechanisms commonly found in many operating systems are missing. However, it is anticipated that AUTOSAR will be extensively used in the automotive industry in the future. AUTOSAR is essentially a specification for a OS, but the concrete implementations of different vendors can vary greatly. It's security meachnisms need to be examined and tested, and it is worth to investigate if there are security mechanisms from more powerful OSs that should also be implemented in AUTOSAR.

Chapter 8 contains a more detailed discussion of all the issues mentioned above.

2.4 Authentication, Interoperability and Mobility

Vehicles are highly mobile and can travel long distances in a short period of time. In Europe commercial vehicles can easily cross the borders of several countries in one day. The high mobility of vehicles requires unified key management for authentication to infrastructures, even beyond country borders. For instance, vehicles should at any time be able to stop for maintenance at a repair shop and connect to their network using wireless communications. These issues are also discussed to a minor extend in Section 6.3.

2.5 The In-vehicle Network

The in-vehicle network spans the whole vehicle and consists of networks of different bus-system technologies, depending on brand and vehicle type: *Controller Area Network (CAN)*, *Local Interconnect Network (LIN)*, *Media Oriented Systems Transport (MOST)*, *FlexRay* and *Automotive Ethernet*. These networks are connected to each other through special *gateway ECUs*, although the internal architecture and number of gateways also vary depending on brand and vehicle type. The Evita project [4] has defined a "use-case" architecture model that describes a possible configuration of a vehicular network.

Traditional security mechanisms cannot directly be used to secure in-vehicle networks due to limitations and constraints specific to vehicles and the vehicle industry:

- Resource constraints of the ECUs, i.e. limited memory and processing power. Complex cryptographic operations or storage of large amounts of data are not possible.

- Cost efficiency. The cost of a typical ECU is in the order of €1 and even a marginal increase of costs is problematic. If the cost of each ECU in a vehicle with 100

ECUs increases by €1, the additional costs for a vehicle manufacturer producing 1,000,000 vehicles per year would be €100,000,000.

- Vehicle Lifetime. It may be in use for at least 10-15 years, preferably even longer. A complicating factor is that the design phase of vehicles is usually very long; vehicles to be released in 10 years are already on the drawing board and many design decisions are already made. Security designs must therefore be modular and sound to survive for such a long time.

Some specialized ECUs, like *gateway ECUs*, may be equipped with additional functionality such as hardware support for encryption, but due to cost constraints most ECUs should preferably consist of standard hardware. This places many constraints on the security solutions that can be implemented in a vehicle.

The use of the EVITA model and how to separate traffic by forming multiple internal networks is further discussed in Section 7.3.

Some vendors are also considering to replace some of the internal wired networks with wireless networks, more specifically Wireless Sensor Networks (WSNs). The main reasons for this consideration are cost and weight savings, which can also translate to less fuel consumption. WSNs present a whole new range of potential security problems. In 2011, SysSec published a state of the art report on sensor network security, which provides a good overview of the topic [7]. Lee, Tsai and Tonguz also provide a good overview of WSNs in vehicle networks [8].

2.6 Product Development and Life Cycle Issues

In this Section issues are highlighted which are related to the product development phase or the relatively long life cycle of a vehicle.

2.6.1 Interplay of Security and Safety

Vehicles should be both safe and secure. A vehicle which is not secure can not be safe, since security vulnerabilities can have a higly detrimental effect on safety. However, this relationship is not necessarily clear cut, standard mechanisms which increase the safety of a vehicle such as redundancy can have a negative impact on security. For instance, a specific hot-standby ECU might increase the attack surface for hackers, since the extra communication and synchronization with the rest of the system can open new security holes. The more complex the vehicle is and the more services are implemented, the more vulnerable the system becomes.

Security functionality must focus on preventing accidents. The functions must be designed to support all safety-critical functionality at the expense of other functions if needed. The system must continue to operate even after a security problem, possibly with some limitations or with degraded service for the driver and passengers.

If drivers start to rely on collision avoidance systems, a silent failure of that system may result in an accident. At the same time, a system that gives too many warnings or behaves erroneously may give bad reputation to the vehicle brand.

Mechanisms implemented for safety, e.g. fault detection mechanisms must be implemented in such a way that they cannot be used by an attacker and result in security problems or evade protection mechanisms. Intrusion Detection Systems (IDSs) and Intrusion Prevention Systems (IPSs) may be used to detect malicious activities and dangerous scenarios. IDSs are discussed more in detail in Section 7.5. But since IPSs also react to what they see, they may be used by attackers to spawn other types of attacks. IPSs may also pose legal problems if they are able to take over the control of a vehicle and bypass the driver's instructions.

2.6.2 Real-time Requirements

Real-time requirements define boundaries on security functionality, both for internal and external communication. In addition, many applications require hard real-time, or near real-time responses to work. Applications, such as collision avoidance systems need to make decisions in a fairly short time to be useful (real-time but not hard real-time requirement), and to make a decision they have to communicate with other vehicles and also be able to verify the correctness of all the messages they receive.

Security functionality must be designed in such a way that real-time requirements can be fulfilled and that it does not disturb other real-time functions in the vehicle. In addition, it is important to protect against misbehaving nodes which try Denial of Service (DoS) attacks, for example by flooding the internal network with traffic so that essential functions fail.

2.6.3 ECU Origins

Vehicle manufacturers often buy finished ECUs from third parties, which comprises both hardware and software. This can make it hard for the vehicle manufacturer to influence the security architecture of specific ECUs. Since a security architecture is only as good as its weakest link, a badly implemented ECU from a third party poses a serious security risk, so vehicle manufacturers need to make sure that the security architecture is well defined and that the security requirements are clearly communicated to the suppliers.

2.6.4 Software Updates and Maintenance

Vehicles have a comparatively long expected lifetime, it is not unreasonable to assume that many vehicles will be in use for 15 to 20 years. The lifetime of security solutions and implemented mechanisms must be equally long. It is worthy to note that this is about the same amount of time that cryptographic solutions have been in use to secure internet communication. Within that time frame, many of those security mechanisms have been broken (e.g., secure hash functions [9]). Since it is not possible to foresee future threats and risks, security mechanisms must be dynamic and adaptable.

The vehicle architecture must allow security functionalities such as cryptographic algorithms, keys, firewalls, etc., to be updated without major software and hardware changes. The SeVeCOM project has therefore suggested an architecture, where security functionality is isolated from the rest of the software and implemented as plug-ins into the network stack to allow easy modification. This approach is described in Section 6.5.

Since it will be necessary to patch vulnerabilities as soon as they are discovered, support for secure remote software updates is needed. This is already common practice in the computer domain (e.g. Microsoft's patch Tuesday), but in the vehicular domain this is still lacking. Vehicles have to be patched not only when they return for scheduled service once every few years, but an internet-based secure software update mechanism is needed immediately.

Different vendors will also have different implementations of similar services, even though they need to communicate with the same devices. Coordination, design, testing and compatibility of such systems will be a major challenge in the future.

2.6.5 Legal Requirements

The software developed for the automotive industry must follow country-specific legislation and requirements. This puts constraints on functionalities and the development processes. It is likely that different countries will have different requirements on security and privacy issues. Law enforcement agencies in different countries may have differing views on what information they need to access. There will also be different organizations handling vehicle identities and issuing certificates, yet all vehicles should be able to participate in future communications and authenticate each other in a safe and secure way. This will be discussed further in Chapter 4.

Chapter 3

Demonstrated Security Threats

The lack of security in today's vehicles has recently been demonstrated by several research groups. By connecting a device to the in-vehicle network, for example through the On-Board Diagnostics II (OBD-II) port, it is possible to send arbitrary commands to the vehicle. Many of the practically demonstrated attacks are performed through this diagnostic interface, which until today has required physical access to the vehicle. However, in the near future, the same attacks will be possible to perform through a wireless connection, with similar results.

3.1 Compromised ECUs can send arbitrary Messages

A team of researchers from University of Washington and University of California have practically shown that infiltrated ECUs in a vehicle can be used to send arbitrary messages to the CAN bus [1]. They have demonstrated this functionality both in the lab and in road tests.

They started by listening (sniffing) the CAN bus to determine how units communicated and what messages were sent. Replay attacks were then trivial to perform. To enhance the functionality in an ECU, they performed reverse engineering by dumping the ECU code over the bus using a third-party debugger. This enabled an attacker to silently "enhance" the functionality of ECUs but still keep its original functionality. They have demonstrated several different attacks: taking total control of the radio (user could not turn it off), produce various sounds in the vehicle (the audio component is used for warning sounds), display arbitrary messages on the instrument panel, open and close doors, honk the horn, disturb engine functions, lock individual brakes, control the A/C, etc.

They also demonstrated that the breaks could be released while driving, making the driver unable to break. These attacks were done locally, i.e. with physical access to the vehicle, but with the correct software, remote attacks with similar results are possible to perform (see below). The team was also able to get access to the high-speed CAN bus by only having a physical connection to the low-speed bus. They identified a system that is

connected to both busses, in their case the telematics unit, and reprogrammed this unit so that it acted as a bridge.

The results are interesting but maybe not surprising since the internal networks and protocols lack all kinds of security, but they clearly show what the outcome could be if a node in a vehicle becomes compromised. Based on these results, Charly Miller and Chris Valasek did similar work for different car brands (see Section 3.4.3), and unlike the previous group, published all engineering details of their hacks.

3.2 Attacks via the Media Player

The same team of researchers from University of Washington and University of California recently also showed that serious attacks can be performed without physical access to the vehicle [10].

A weakness in the media player was used to gain access to the local CAN bus. The media player they tested came from a large third party supplier and it plays CDs and also accepts MP3 and WMA files. Media players in vehicles normally have access to the CAN bus, for example to be able to change the sound level when the vehicle's speed changes. Using reverse engineering, they discovered that it was possible to do a buffer overflow attack when playing WMA music. The weakness allowed the attackers to create a CD which contained specially crafted music that the player plays perfectly, but also silently performed a buffer overflow attack which can send arbitrary commands on the CAN bus. It is not hard to imagine that music containing such viruses would be popular to distribute on the internet. The telematics unit also contained vulnerabilities in its cellular communication protocols opening it up for remote internet attacks.

The attacks show the necessity to have good security practices in place when designing software for ECUs and other equipment connected to the internal networks. In this case, the media player and the telematics unit in the vehicle had traditional buffer overrun bugs, allowing arbitrary messages to be sent on the internal CAN bus. In the future, we can expect that many ECUs and subsystems like these contain software implemented by third parties, about which the vehicle manufacturer have limited knowledge and no possibility to know all design details or to be able to evaluate the implementation.

3.3 Attacks via Wireless Tire Pressure System

All vehicles manufactured in the U.S. after 2007 are required to have a Tire Pressure Monitoring System (TPMS) installed. Rouf et al. [11] have demonstrated an attack where the wireless communication (RF transmitter) between the vehicle and the sensors in the tires are compromised.

Each tire sensor has a unique 32 bit identifier to prevent vehicles from displaying messages from other vehicles. The radio communication between the tires and the vehicle turned out to be unprotected and reached around 40 meters with low-cost antennas and amplifiers. They also showed that remote spoofing of messages was possible, and all

messages with correct IDs were unconditionally accepted, triggering warning messages for the driver. The equipment used was available on the market for around $1,500.

During their tests with spoofed messages, they also managed to completely crash the ECU receiving the tire-pressure messages, and the only recovery possible was to return the vehicle to the repair shop and have the ECU replaced.

They also concluded that the 32 bit identifiers used to uniquely identify each tire pressure sensor, can be used to track vehicles and therefore creating privacy problems for the owners.

3.4 The Car as part of the Internet of Things

Since 2009 many vehicles have a built-in sim card module to offer customers new services, such as real time traffic information, automated emergency call in case of an accident, remote diagnostics, or control via smartphone application (unlocking the vehicle, or turning on the pre-heating). The vehicle as part of the internet introduces new security flaws, as it provides remote accessibility.

3.4.1 100 Cars disabled Remotely

In March 2010 media, including *wired.com*, reported that more than 100 cars were disabled remotely. Until the cars were fully paid for, the cars were controlled by the car dealer and contained functionality to be remotely disabled if the customer slipped with the monthly payments. 1,100 cars were reported to have this functionality.

This day, a former disgruntled 20 year old employee used a fellow employee's account to log in to the dealership's computer system and disabled more than 100 vehicles for their customers. The ignition system was disabled and he managed to honk the horns in the middle of the night. The only way the customer could turn off the horn was to remove the battery.

This security problem again shows the risk of installing third party applications in vehicles. There may have been limitations to what the system could have done, but we have also seen that the vehicles lack protection if an ECU or a connected system wants to send arbitrary messages.

3.4.2 Unlock of a vehicle remotely

In 2015 a vulnerability has been found that allowed an attacker to unlock a vehicle remotely. The car manufacturer offered a remote service to perform certain control and infotainment commands via a smartphone. An analysis has shown that a man-in-the-middle attack can be used to gather the unencrypted *unlock* message from the mobile phone by mimicking a cellular base station and replaying this message. In this particular case, the car manufacturer was able to patch 2.2 million vehicles having this security flaw with an over the air software update, as this vulnerability didn't involve the hardware modules [12].

3.4.3 Remote Compromise of a Jeep Cherokee

In 2012 Miller and Valasek started their research, which was funded by the Defense Advanced Research Projects Agency (DARPA), to root out vulnerabilities in vehicles. Their first hack required a physical connection and allowed them to control the vehicle and even disable the brakes. The demonstration was performed on a Toyota Prius and a Ford Escape. Miller and Valasek focussed on attacking the system via a physical connection, because of the already published hacks to access a vehicle remotely. Their findings have been published in 2013 [13].

In 2015 Miller and Valasek presented their results in remotely hacking a Jeep Cherokee at blackhat USA 2015. They started their investigation on possible vulnerabilities with a 2014 Jeep Cherokee. Their first discovery was that the radio was connected to both of the vehicles CAN busses. The head unit (radio) has not been developed by FIAT Chrysler Automotive, it was produced by one of the company's supplier [14].

Further investigation showed that the WiFi password of the vehicle was generated automatically based on the time when the head-unit of the vehicle was started the very first time. The knowledge of the production year and the month can narrow down the number of combinations to 15 Millions. By taking also the fact into account that the vehicle was likely started the first time during the day, one is able to reduce the possible combinations to about 7 millions. Nevertheless, the researchers found out, that the WiFi password is set before the system has set the correct time. In their case it was the 1^{st} January 2013 at 00:00:32. This fact reduces the possible number of combinations dramatically and allows a nearly instantaneous access to the vehicle's WiFi. The implemented password generator would have forced the attacker to follow the target for about one hour to have a steady WiFi connection when performing the brute force attack. However, knowing that the generated password uses an incorrect system time reduces the time for the attack tremendously. This vulnerability emphasizes the importance of security tests in third-party devices [14].

The access to the vehicle's WLAN network allowed the researchers to further investigate the D-BUS interface, which has shown that no authentication was necessary. Changing the volume, fans and even the display was possible with lua scripts due to the open D-BUS interface of the Uconnect system [14].

Miller and Valasek were also able to jailbreak the Uconnect system by switching the pendrive while the vehicle reboots. Moreover, the authors highlight, that the jailbreak of the system was not required to access the vehicle's D-BUS. The use of a femtocell allowed the researchers to access the vehicle from even farther away. Thus, vehicles that do not have enabled the vehicle's WiFi service are vulnerable as well. Surprisingly it turned out, that the D-Bus port was not only opened to the WiFi and the closest cell, it was opened to the entire Sprint network. This has the result, that all vehicles were accessible by any device connected to the Sprint network. A scan of certain IP ranges, that are used by Sprint for vehicles, with an open D-Bus port (in this case port 6667) and a certain

reaction via telnet revealed that 16 car models across the United States of America have been effected by this vulnerability [14].

To perform cyber physical actions remotely, it was necessary to flash the firmware of the V850 chip that communicates with the ECUs via CAN bus. Reverse engineering the firmware of the Uconnect system's chip and the V850 chip in order to find a way to remotely upgrade the firmware of the V850 chip took the researchers several weeks [14].

Nevertheless, Miller and Valasek were not able to perform steering or braking while the vehicle was driving more than 10 mph. This restriction was caused by the ABS and Parking Assist Module (PAM) ECU that turn off some of their functionality in case they receive contradicting messages (the valid/correct and malicious messages). For example, the ABS ECU disables the collision prevention entirely when receiving conflicting messages. The researchers had to put the ECUs which are sending the correct messages into the diagnostic mode, what is only possible below 10 mph, to be the only one sending a certain message type. A discussion about the correctness in terms of safety and security of this mechanism encountered in the Jeep Cherokee is left open. This model is not a self-driving vehicle and thus the driver has to be alert at any time, such as in the case when the collision prevention is suddenly disabled. However, the higher the level of automation of the vehicle, the more robust and fail-safe in terms of safety and security the vehicles have to be.

This research has caused a recall of 1.4 million vehicles for updating the software of the vehicle. Moreover, it is shown that the vehicle as part of the internet of things involves also network providers. The security of the vehicle depends also on the devices/modules from the suppliers. In this case, the vulnerability of the head unit (Uconnect system) allowed the attacker to access the vehicle's system from all across the US [14].

In 2016 Miller and Valasek have revealed a vulnerability of the in-vehicle network of a Jeep Cherokee that was patched against the vulnerability they have published the year before. The focus of this hack was the full control over the vehicle, thus they connected a notebook directly on the ODB-II port. The researchers were able to bypass safeguards that prevented the full control over the vehicle. Their approach was to not only compromise one ECU in order to send control commands, they attacked those ECUs that were sending the correct messages the researchers were sending. This causes the system to only receive their "wrong" messages sent over the CAN bus. Disabling the safeguards was achieved by paralyzing these ECUs by putting them into "bootrom" mode. Miller and Valasek claim that this attack is still possible and that it is only a matter of time that attackers find a new vulnerability to access the vehicle over the internet [15].

3.4.4 Web Portal as Target

In the mid of 2016 the security researcher Benjamin Kunz Mejri of Vulnerability Laboratory has found two vulnerabilities in the web interface of a car OEM that allowed the attacker to access the configuration of other vehicles. Accessing and manipulating the configuration of another vehicle was possible due to a vulnerability in the session

management that allowed to bypass the secure validation [16]. This attack allows the attacker to access and manipulate the playlists, e-mail accounts, travel routes, and to unlock or lock the vehicle. The second security flaw is a client-side cross site scripting web vulnerability, that allowed the attackers to inject malicious code [17].

3.4.5 Accessing the CAN bus via WiFi

Researchers from Tencent's KeenLab security team have successfully hacked a Tesla car in 2016. The researchers created a WiFi network that is typical for Tesla stores and exploited a vulnerability of the web browser, which is based on the WebKit framework. This bug allowed the researchers to run malicious code, reprogram the firmware of the gateway that has access to the CAN bus, and in the end to control the vehicle. Tesla not only patched the WebKit framework, the company also introduced *code signing* in order to avoid the firmware installation of unsigned code [18]. This example also demonstrates that third-party software or libraries might lead to new vulnerabilities that need to be tested beforehand or patched in a timely fashion.

3.4.6 Smartphone vulnerabilities

Promon, a security company based in Norway, has shown that it is possible to hack a Tesla car with a smartphone as entry point. They created a WiFi hotspot of a popular burger restaurant and trick the user into installing an application in order to get discount. The installed application is used to control the Tesla application, so that it sends the attacker the user credentials when logging in. With the username and password of the Tesla owner, one was able to track the car, unlock it and drive it [19].

The researchers highlight, that Tesla didn't consider best practice examples for security in their smartphone application, which lead to an easy and fast extraction of the user credentials. Furthermore, it is important to clarify that it is also possible to trick the user into installing a custom keyboard to get the necessary information. The users need to realize that the smartphone contains important information, such as information from banking apps, and that they have to use it more carefully.

Chapter 4

Privacy and Legal Implications

Privacy has many different definitions, and the notion of privacy can therefore be ambiguously interpreted. To avoid such problems, we will first discuss different privacy definitions. Two well-known definitions are "privacy as secrecy" by Posner [20] and "privacy as control" by Westin [21]. *Privacy as secrecy* concerns an individuals' right to conceal certain things about themselves, whereas *privacy as control* is the right to control what information about oneself is communicated to others. In this chapter we will consider *privacy as control* as our notion of privacy, meaning that any information that was not intended to be publicly released will be deemed sensitive personal information.

Another recurring disagreement within the area of privacy is whether or not encryption should be considered a privacy-preserving technique or not. Li et. al [22] summarize the differences between privacy and confidentiality nicely, by pointing out that confidentiality just requires secrecy, whereas privacy requires both secrecy and utility of data. Consequently, Li et. al call attention to order-preserving encryption (OPE) as being the encryption technique that is most similar to privacy, as it allows the order of records to be inferred even in the encrypted form, thus also providing utility. As a result, confidentiality achieved by encryption will not be considered a privacy-preserving technique in this chapter.

4.1 Privacy Challenges

A major challenge with privacy is being able to determine under what conditions data can be considered private. One reason for this is the *inference problem*, where the presence of *auxiliary data* allows additional data to be inferred. The inference problem is commonly occurring in the wild, and next we will provide some notable examples.

One famous example is when the governor of Massachusetts decided to release anonymized medical records, where names and social security numbers of patients had been removed. It later turned out that users could be re-identified by cross-referencing people from another source of information, in this case a voter registration list [23]. In a similar manner, users from Netflix, an online video and tv streaming service, could be

re-identified even though their data had supposedly been anonymized, by for example cross-referencing the anonymized data with public data from the internet movie database (IMDb) [24]. Another notable example is the AOL debacle, where a journalist managed to re-identify and interview an AOL user, who was supposed to be anonymized in a public data set containing search queries [25].

Related to the vehicular domain, a similar re-identification was possible when insurance companies promoted privacy-preserving insurances for cars. In this case, privacy-preservation was achieved by removing GPS data and only collect a time series of the car's speed. While this might at first seem harmless, Gao et al. [26] showed that the combination of driving speed, timestamp and home address, which the companies also had access to, of the drivers was enough to deduce the location of a vehicle by comparing driving patterns to road maps. Comparably, Lee et. al [27] showed that more information that intended can be obtained from seemingly harmless data by demonstrating that the RPM together with the throttle position sensor (TPS) can be used to deduce the fuel consumption of a car.

All the aforementioned cases have one thing in common, they naively release data without considering what can be deduced from it in the presence of auxiliary data. Identifying what attributes lead to information disclosure through inference is difficult. Therefore, anonymization of data is a promising approach to being able to store and process data without violating users' privacy by falling victim to the inference problem. However, guaranteeing anonymization is difficult, especially if data is high dimensional and high frequency, as is the case for car data. Thus, for connected cars, choosing what privacy-preserving technique to be applied to the data is not trivial. What technique it suitable for what type of data is further investigated in the ongoing BAuD II project[1].

4.2 Privacy Models

Privacy models are used for providing privacy after or as it is being gathered, which enables privacy-preserving analyses to be run. There are two main approaches of privacy models, those that are designed for privacy-preserving data publishing (PPDP) and those that are made for privacy-preserving data mining (PPDM). The PPDP models focus on releasing whole sets of data, often in the form of high-dimensional tables, and PPDM models, represented by differential privacy in this chapter, that typically release anonymized query answers.

4.2.1 Differential Privacy

Differential privacy [28] is a definition that provides strong, provable, privacy guarantees. The idea behind differential privacy is that the contribution of one single record to a database query should be negligible, meaning that the presence or absence of that record will be unnoticeable. In other words, the privacy risk to an individual participating in

[1] http://www.vinnova.se/sv/Resultat/Projekt/Effekta/2009-02186/BAuD-II-Storskalig-insamling-och-analys-av-data-for-kunskapsdriven-produktutveckling/ (2017-02-28)

a database will be indistinguishable from that individual not participating. In practice, differential privacy is often enforced by applying Laplacian noise to a numerical query answers, but there are other privacy mechanisms that can be used.

The main selling point of differential privacy is that its guarantees holds in the presence of auxiliary data, including data that will be released in the future. Consequently, differential privacy is both *forward and backward proof* against auxiliary data. Furthermore, differential privacy is also *composable*, which means that queries can be constructed in such a way that their results can be combined while still keeping the privacy guarantees intact.

While differential privacy is promising, it is challenging to apply correctly, which results in either incorrect assumptions that violate differential privacy, or results with too low utility.

4.2.2 Syntactic Privacy Models

The syntactic privacy models all stem from *k*-anonymity, which was invented by Samarati and Sweeney in 1998 [29]. *k*-anonymity provides privacy by grouping together k records whose *quasi-identifiers*, a chosen set of non-sensitive attributes, are all the same. Thus, each record is hidden among $k-1$ records with the same quasi-identifiers. *k*-anonymity does not dictate how each group of k records is created, which means generalization and suppression can be performed before the grouping is carried out.

For example, Table 4.1 shows a table with raw data, that then is turned into a table satisfying 2-anonymity in Table 4.2. Notice that one attribute, license plate, is suppressed in this example, as it is a unique identifier. The quasi-identifier in this case is the attributes manufacturer and model.

License plate	Manufacturer	Model	Speeding
ABC123	VCC	V70	Yes
CBA123	VCC	V70	No
ABC321	VCC	XC90	Yes
CBA321	VCC	XC90	No

Table 4.1: Raw table, where the sensitive information is if a car has been speeding or not

Manufacturer	Model	Speeding
VCC	V70	Yes
VCC	V70	No
VCC	XC90	Yes
VCC	XC90	No

Table 4.2: Table that enforces 2-anonymity

Since *k*-anonymity provides no privacy guarantees, the model has been subject to attacks. One of these attacks, the similarity attack, where all sensitive attributes share

the same value, led to the invention of a successor called *l*-diversity [30]. An example of the similarity attack is shown in Table 4.3.

Manufacturer	Model	Speeding
VCC	V70	Yes
VCC	V70	No
VCC	XC90	Yes
VCC	XC90	Yes

Table 4.3: Table that enforces 2-anonymity, but which has the same value for the sensitive attribute in one of the groups

Most recently invented is *t*-closeness [31] that solves some of the problems with *l*-diversity.

While the syntactic privacy models are easier to apply than differential privacy, none of them provide any formal privacy guarantees. They also do not support composability, so any release of a data set is one-shot, meaning that in case the result does not prove to be satisfactory, no more information can be retrieved from the raw data without violating the privacy model. Another problem is that the implementer must choose the quasi-identifier, which means the implementer decides what attributes are sensitive. To successfully do so, the implementer must also be aware of what information can be inferred from the other published attributes.

4.3 Private Communication

While communicating with other cars or infrastructure, privacy might also be desirable. Since privacy models are not applicable here, we explore grouping through group signatures and obfuscation through pseudonyms.

4.3.1 Group Signatures

When using group signature, messages are being signed with a group identity. Any member of the group is able to sign messages, and any other entity can verify the authenticity of the signature. The authentication is limited to the proof that the message has been signed by a member of the group, not by whom it was signed. With this method one is not able to identify the particular individual which has sent the message. Group signatures may be promising but more work is needed before they can be used in practice [32].

A possible attack on this scheme is if all entities but one in the group are colluding, which makes it possible to deduce who the signature came from. In other words, this scheme is private under the assumption that adversaries are non-colluding.

4.3.2 Pseudonyms

With pseudonyms, the real identity of the object is hidden from, for example, other vehicles and road-side objects. Pseudonyms require that the system supports short-lived public key certificates. These pseudonym certificates must be issued by Certificate Authorities (CAs) and are therefore not trivial to introduce [33]. One possibility is to equip vehicles with a number of pseudonyms when it is in contact with the CA and give it certificates that can be used for a short while.

The pseudonyms must be signed by a CA to allow an authorized entity (an authority) to track the real identity of a vehicle, if needed. The system must support non-repudiation to make it impossible for an entity to deny having sent a message, for example if a vehicle fabricates warning messages to other vehicles. The use of many pseudonyms makes caching of certificates in vehicles less effective.

A similar privacy issue is that the communicating vehicle's MAC address (and IP address, if used) can be used for identification since it is unique for the vehicle. When a vehicle decides to use another pseudonym certificate for identification, the addresses need to change as well. The solution can be to use random addresses from a pool earlier given to the vehicle, similar to how pseudonym certificates are generated and used.

The main problem with pseudonyms is that they do not hide communication patterns, which in themselves might reveal additional information. This problem increases with time, as patterns become more visible with more data, and changing pseudonyms is therefore recommended. Another problem is that the mapping between the true identifier and the pseudonym must be stored somewhere secure, or else privacy will be breached.

4.4 General Data Protection Regulation (GDPR)

The General Data Protection Regulation (GDPR) is a new European law that applies from the 25th of May 2018 [34]. The GDPR grants individuals more control, as well as easier access to their data. Since the GDPR covers "personally identifiable data", anonymized data is not included in the GDPR. Until a real case is brought up in court however, the legal interpretation of "anonymized data" is unknown.

Following is a list of some of the new rules from the GDPR [35]:

- "Right to be forgotten"
- Easier access to one's data
- Data portability
- The right to know when one's data has been hacked
- Data protection by design and by default
- Stronger enforcement of the rules

GDPR especially promotes privacy-preserving techniques such as anonymization, pseudonymization and encryption of data.

Chapter 5

Common Network Security Threats

The internet is a constant source for malicious traffic [36], and since the internet-connected car will have a public IP address, it will like any other internet-connected computer system be a target for such malicious traffic. Security mechanisms similar to what we use to protect traditional computer systems and networks are needed. They must be adapted to the specific requirements and limitations vehicles have and still be able to meet the high security demands of safety-critical systems.

Most research in the area has focused on algorithms and what messages need to be exchanged for various applications, such as for control systems for platooning, how to implement cooperative lane-change support, display messages about approaching dangers, etc., but security is often treated as a future issue [37, 38]. Standards for external V2X communication are emerging. IEEE just recently published the security architecture used in WAVE; the IEEE 1609.2 [39]. To conclude, we can easily identify a plethora of threats, but available and usable security mechanisms are to a large extent missing and, as we will see, much more work is needed in this area.

5.1 Communication Threats and Problems

All protocols, at all levels, must ensure they have proper protection against several types of attacks.

- *Eavesdropping:* attacker reads (copies) data from the network. For wireless communications, sensitive and confidential data needs to be encrypted. An example of confidential data can be the software (firmware) for the ECUs.
- *Deletion and modification:* data is dropped or modified during transmission. Application-level protocols must either have their own detection mechanisms or ensure that the underlying network protects the data. Modification of traffic can be done by a man-in-the-middle attack, for example by abusing link-level protocols and changing the traffic routing.

- *Injection and data origin spoofing*: new traffic is injected into an ongoing session either by a man-in-the-middle or by a remote attacker. In a repair shop, an attacker may insert additional diagnostics messages into an existing session in order to steal data, change settings or the configuration of a vehicle, or possibly even update the firmware with new "enhanced" functionality.
- *Impersonation*, also called identity spoofing. The identity of other trusted users or devices may be spoofed with similar results as for traffic injection.
- *Recording, replay and delay:* data from older sessions are reused. It could contain old authentication information or the attacker could resend information that was intended to be used during other circumstances, at other locations or reuse data that should have been valid only during a short time period. Even authentic data which is immediately relayed over the internet to another location and replayed, could cause confusion and problems.
- *Denial of Service attacks:* often done by flooding networks or by using known vulnerabilities that cause nodes to crash or exhaust their resources.
- *Malicious traffic:* Malformed packets that should not normally be seen on networks but can cause systems to malfunction, crash or execute arbitrary code.

5.2 Protocol Vulnerabilities

There are many protocol stacks implementing link, network, and transport services in a modern vehicle which can potentially be exploited. In addition, not only attacks against the low-level network stack are possible, but all application level protocols that are introduced into the vehicle are potential targets for attacks.

Manipulation of data from application layer protocols may also result in severe security problems, for example enable modification of the functionality of an ECU.

There will be many application-level protocols, some based on TCP/IP and communicating in traditional client-server manner, and other use broadcast communicating to reach only objects around the car (V2X). Some applications will take care of security themselves, others will rely on security offered by the communication technology itself, for example for encryption and authentication. ETSI has been mandated to standardize V2V communications within the European Union [40]. Future standards such as their ITS Station architecture [5, 41, 42] will therefore likely dictate security requirements for nodes participating in V2X communications, but it will not help developers with how security functionality should be handled within the vehicle and how it may be implemented in various applications.

Unauthorized manipulation of traffic to vehicles may also result in unwanted behavior, for example to enforce non-existing speed limits, to avoid empty but seemingly congested roads, announce lane-changes without the driver having any such plans, cause problems with car platooning and in general disturb functions in and around the car. A more

dangerous scenario is an attack that sends spoofed messages to ECUs, for example that orders full speed ahead and disabling the break ECUs when the sensors in the car detect pedestrians in the way.

5.2.1 Attacks against Core Protocols

All protocols in the network stack can be attacked. The internet-connected car uses the IP protocol and therefore all Core protocols (DHCP, ARP, ICMP, DNS, TCP, UDP) as well as link-level protocols (802.11p, WLAN, cellular 3G/4G, etc.) face similar threats as other systems do. Application level protocols are unique for this domain and will be obvious targets for attacks and must be designed to be both secure and robust.

The V2X communication protocols (WAVE, IEEE 802.11p, WSMP) can also be targets for attacks, as well as Bluetooth, DSRC and RFID communications.

Attacks against core protocols have existed for decades and many tools exist that can be used by attackers but also by developers to perform their own penetration tests to make sure that at least all old and well-known vulnerabilities are handled properly. Historically, there are many well-known core protocol attacks, for example the Land attack (victim's address present in both source and destination fields), Ping-of-death (oversized IP datagrams), and header exploits (e.g. the length specified in protocol headers differ from the actual length) which may cause various problems from crashes to execution of arbitrary code.

This list of known problems is short enough to allow developers to test their implementations against almost all of them. However, history has shown that many new implementations do not take these known attacks into consideration, thus they re-appear in new implementations and cause systems to fail. Examples include Microsoft's network stack in the beta version of Windows Vista where they failed to perform such tests [43], and in Smartphone operating systems where a single link-level datagram can make the device freeze completely and only a removal of the battery makes it functional again [44]. These attacks mainly target the communications unit of the vehicle, although a failure in it may result in a denial of service, or worse, that malware is installed that can send messages fabricated remotely by attackers, on the internal busses.

Many of the threats to vehicles can be analyzed using the same methods and tools as are used to secure other internet-connected systems, and they should be used also in this environment to test the robustness of the implementations. Lang et al. [45] provide an interesting discussion of the security implications when the vehicle is connected using an IP-based network. Nine "hypothetical attack scenarios" are suggested based on attacks known from ordinary IT systems, i.e. attacks on the communication protocols, malicious code, and social engineering. Each scenario was analyzed with respect to confidentiality, integrity, availability, authenticity, and non-repudiation. Also, an attempt to quantitatively estimate the impact on safety was conducted and for each of the scenarios, a SIL value (see Section 13.2) was proposed.

Chapter 6

External Communication

Vehicles already provide many interfaces to external devices. A service technician is able to connect a diagnostics device to the vehicle to check the internal status and to update the firmware of the ECUs. Connectivities, such as Bluetooth, used to connect to the vehicle's infotainment system are also present. Exchanging information via V2X communication is a promising technology to increase the road safety and efficiency of the vehicles. Other use cases that require external communication are the automated payment of road tolls, and diagnostics over the internet.

6.1 Communication Technologies

Many different communication technologies are likely to be used for various communication services. Some technologies, like DSRC and Bluetooth are very local and the objects need to be located closely to the vehicles. Other techniques, such as 3G and similar technologies are used for accessing the internet to and from the vehicles.

6.1.1 Technologies and related Standards

DSRC, Dedicated Short Range Communications is a general term often seen in V2X documents but the meaning of the term DSRC has changed over time and may today mean RFID communication, WLAN communication or any other of several different short range communication techniques. In the vehicular domain, DSRC is synonymous with the use of the 5.9 GHz frequency band which is dedicated for V2X communications. Standardization is going on in this domain and in United States the IEEE 1609 WAVE (Wireless Access in Vehicular Environments) protocol which uses the IEEE 802.11p link-layer protocol will be used. In Europe the ITS-G5 which is also on the basis of 802.11p, but with some amendments towards European requirements is being standardized to be used for vehicular ad-hoc communications.

IEEE 802.11p is a technology based on 802.11a but has some special functions making it more suitable to short-lived ad-hoc communication. Higher layers are expected to follow the IEEE 1609 WAVE standard, which covers layer 3 to 7, where both short

broadcast communication as well as TCP/IP is supported. These protocols and their security features are further discussed in Section 6.1.2. The 802.11p standard supports up to 27 Mbps using 7 dedicated channels in half duplex mode and communication range is in the order of 300 meters. 802.11p does not explicitly address security and encryption (like WPA and WPA2 which is present in 802.11a) and it must therefore be addressed by upper layer protocols. Security was omitted because of the short-lived communication patterns and problems with authentication of users on lower protocol layers. Instead of spending valuable time computing crypto-keys and exchanging messages with a base station, vehicles should be able to quickly create ad-hoc networks and exchange signed messages directly with as little overhead as possible.

ITS-G5 is the name of a collection of European standards for communication in the 5 GHz band, based on IEEE 802.11p. They specify the functionality of the physical and data-link layer used in the ETSI ITS communication (ITSC) architecture to ensure cross-border interoperability. More details are provided in section 6.1.2.

WLAN technology, i.e. traditional IEEE 802.11a,b,g,n communication can also be used and offer internet access to traditional client-server applications, for example by car owners who want to connect the car's multimedia system to their home network. The car manufacturers and third party software vendors may also want to use WLAN technology for remote diagnostics and software updates while the vehicle is parked.

Cellular networks will primarily be used for client-server based applications and used where no WLAN networks are present. It will also be used for safety-related services, such as calling for help if internal sensors have detected a crash and by many other types of applications when road-side devices cannot be used. Cellular communication can also be the communications method used to check the validity of certificates by consulting on-line revocation (CRL) lists.

Bluetooth technology is used today to connect mobile phones to the multimedia system, primarily for hands-free operation. This may seem harmless, but the multimedia system is connected to the internal CAN bus, and if compromised, arbitrary messages may be sent. Most mobile phones are at the same time connected to the internet and may function as bridges between the internet and the internal vehicle network. In the future, Smartphones can also use Bluetooth communication to offer many other services to vehicles.

NFC, Near Field Communication, such as RFID cards are today used for driver identification and can in the future offer more functionality. Vehicles communicating with other devices using Bluetooth, NFC, etc., are often called *V2M* (vehicle to mobile communication) and are sometimes, but not always, included in the *V2X* concept.

Global Positioning System (GPS) and RDS radio communication. RDS information is used to receive broadcast messages about traffic conditions and road work. GPS and RDS technology is in place today, but security problems may arise if ECUs within the car always trust the transmissions and services offered to vehicles depend on the correctness

of the data. A driver may, for example, change his GPS position in order to avoid road tolls and other fees.

6.1.2 Broadcast V2V and V2I Communication

Many V2V and V2I services will use broadcast DSRC communication. Messages can contain information such as a vehicle's location, speed, directions, maneuvers such as intention to brake, change lane or pass an intersection, etc. These broadcasted messages are used to spread awareness between vehicles close to each other.

Wireless access in vehicular environments has recently been standardized by both the IEEE to be used in United States and the ETSI to be used in Europe. In the U.S., the IEEE 1609 WAVE standard supports V2V and V2I communications using the DSRC 5.9 GHz band dedicated for the intelligent transport system (ITS). WAVE standardizes both V2V and V2I communications and describe data exchange, security,and service advertisement between communicating parties and is intended to be a framework for application developers when implementing services. The IEEE 1609.2 standard which describes the security for the WAVE communications is an active standard now. Figure 6.1 shows an overview of the protocol hierarchy and the different protocols used at the different communication layers.

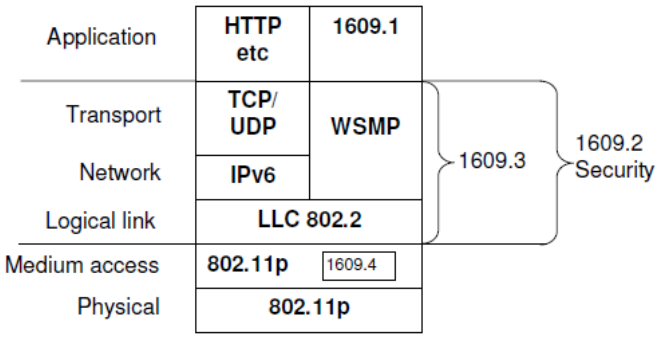

Figure 6.1: WAVE = IEEE 1609 + IEEE 802.11p

The WAVE standard relies on the **IEEE 802.11p** protocol for the physical (PHY) and link (MAC) layers and covers layer 3 to 7. The WAVE standard governs modes of operation in the DSRC band including architecture and resource management (1609.1), management, security services and message formats (1609.2), and network services (1609.3). A new protocol, the *WAVE short message protocol* (WSMP) has also been specially designed for broadcast V2X communications. In addition, two special service channels allocated for safety-critical applications also support IPv6.

The ETSI ITSC architecture standardizes the functionality contained in ITS stations and it follows the principles of the OSI layered architecture extended by amendments towards European ITS applications. Figure 6.2 shows the ETSI ITSC reference architecture.

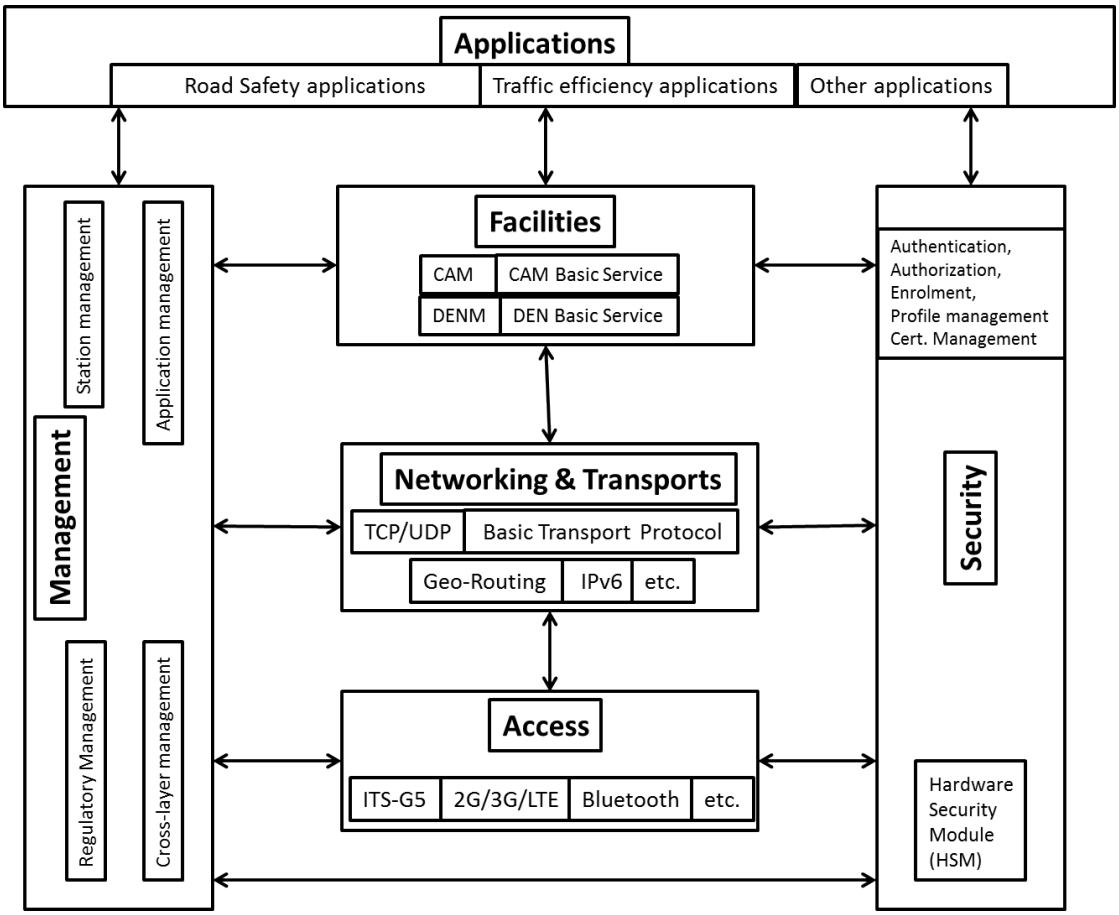

Figure 6.2: ETSI ITSC reference architecture

The ETSI ITSC architecture is specified in ETSI EN 302 665. The Table 6.1 shows some details about the ETSI ITSC architecture and the related publications.

6.1.3 Vehicular ad-hoc Networks, VANETs

VANETs are often considered to be a subset of MANETs, Mobile ad-hoc networks. The most important differences are the short-lived communication patterns in VANETs and that it is unlikely that the same nodes forming a VANET will ever meet again. Thus, vehicles will frequently participate in new dynamic ad-hoc networks with participants they have never seen before, and it is therefore not that meaningful to store or cache any credentials or information to speed up reconnection to the same VANET again. Many VANET applications such as collision avoidance systems also have hard real-time requirements for the communication.

The lifetime of a VANET can be very short. Consider a vehicle approaching a road-side object in 100 km/h. If we assume that communication can take place in a 100m

Layer	Description	Related ETSI Publication
Access	Corresponds to the Physical and Data Link layers of the OSI model	EN 302 663 | ES 202 663 TR 102 960 | TS 102 917 TS 102 916 | TS 102 862 TS 102 861 | TS 102 792 TS 102 724 | TS 102 708-2
Networking and Transport	Corresponds to the Network and Transport layers of the OSI model	TS 102 636 | EN 636-2 TS 102 985 | TS 102 870 TS 102 871 | TS 102 859 TR 103 061
Facilities	Corresponds to the Session, Presentation and Application layers in the OSI model. It provides Application support, information support and communication/session support	TS 102 894 | 102 637 TS 102 869 | TS 102 868 TS 102 863 | TR 103 061
Management	Manages the installation and configuration of ITS-S applications, responsible for regulatory management, cross-layer management, etc.	TS 102 723 | EN 302 665
Security	Providing security services, such as certificate management authentication/authorization, cryptographic functionalities, etc.	TS 102 731 | TS 102 943 TS 102 942 | TS 102 941 TS 102 940 | TS 103 097
Application	Road safety, Traffic efficiency and other applications	TR 102 638

Table 6.1: ETSI ITSC architecture layers and related publications

radius from the object, they can only communicate during 7 seconds which includes time for network discovery and possible authentication procedures take. The lifetime of communications between cars driving in opposite directions may be even shorter, and it may therefore be necessary for other vehicles to forward messages to extend the communication range.

Some messages need to be repeated or forwarded by the nodes in the network to reach all participants, possibly with a delay to allow vehicles to move to increase the communication range. Some messages like emergency messages should be spread rapidly and reach as many recipients as possible in a short time, but this can result in network flooding if lots of nodes begin to repeat the same message. Therefore, new routing protocols are needed that are adopted to this situation.

One interesting technique is to allow vehicles (nodes in the communication network) to store information and at a later time and position, retransmit the data, i.e. data is carried from one location to another by vehicles. This will increase communication range for messages, but it also results in new routing problems. Several methods for how this can be done have been suggested, for example using group communication techniques. All these requirements and restrictions make many algorithms developed for MANETs less useful and new methods for communication are therefore under development.

6.2 The ITS Station

ETSI has standardized an "ITS station" which is a standardized communications node intended to be used by all V2X communication systems [41]. The use of such a standardized platform is to make it possible to certify highly specialized communication nodes and make them highly resistant against failures, including security-inflicted problems. The ETSI ITS station reference architecture describes the functionality and tries to standardize how V2X communications should be performed. The standard specifies the architecture for hosts, gateways, routers and border routers. The (ITS-S) gateway is a bridge (or protocol converter) connecting two protocol stacks at layers 5 to 7 and in order to achieve this, it requires two full protocol stack implementations. It covers functionality like classification, prioritization and channel assignment and maintenance which can be requested by applications.

Some protection against DoS attacks originating from vehicles can also be part of the functionality of the ITS station, if the hardware has some means to filter what can be transmitted by individual ECUs. For example, there could be hardware-enforced limitations to what can be transmitted and how many messages can be transmitted during a certain time period. For such functionality to be fully trusted, it is essential that it cannot be modified through software updates or by any type of failure of the ECU, i.e. that (at least parts of) the communications module is fully isolated from the rest of the ECU.

Hamida et al. described in [46] a list of threats to an ITS station. These threats are: the availability, identification and authenticity, confidentiality and privacy, integrity and data trust, and non-repudiation and accountability. The following Sections cover a discussion about these threats. Moreover, privacy in vehicular systems is discussed in Chapter 4.

6.3 Vehicle Authentication, Message Integrity and Confidentiality

There are many situations where proper authentication of the communicating parties is needed. In cooperative driving the vehicles aim to solve a problem by interacting with each other. Thus, the verification of the other entity's identity is the base for further message exchange and interaction.

Authentication is an important security function needed in most types of communication and the use of public key signatures has been proposed to simplify deployment of a large authentication system. Several organizations can help in distributing certificates in their local region. However, the deployment is still far from trivial and vehicles may have to be constantly connected to a central system to be updated about revoked certificates, e.g. to check or download Certificate Revocation Lists (CRLs).

Although certificates have been proposed and are believed to be the way to implement authentication [47], there are still problems to be solved with this technology [48]. There will be lots of issuers of certificates and vehicles are not limited to country borders. Traditionally each country has had their own national organization issuing license plates, but it is not clear if these organizations are willing to take over the role of issuing digital certificates and work with key generation and distributing CRLs. In the US each state is responsible for its vehicles and vehicles are frequently crossing the borders. The frequency for replacing keys and what (limited) lifetime they should have is still open for debate. Similar problems exist with CRL distribution lists and their scope.

Message Integrity & Proof of data correctness is often more important than identifying the sender. Data can be forwarded by many devices. Some data can be trusted based on its contents given the conditions for when and how it is sent, even if it is sent by a node whose identity cannot be properly verified.

Some applications, as mentioned in Section 6.3.3, may want to communicate in closed groups. Traffic should be encrypted and only certain vehicles are allowed to participate, for example those subscribing to a particular service or those of a specific vehicle brand. Group communication can be accomplished by getting a temporary symmetric key by an RSU when entering a region. Only authenticated vehicles get the key and are able to communicate. Techniques like this may lessen the burden on vehicles to authenticate all other vehicles, something that can be important when reaching highly populated areas.

Authentication is an important building block when implementing security, and there are many situations where proper authentication and identification of traffic and communicating parties is needed:

- *Identification of V2X traffic and/or entities*: to be able to identify RSUs and other entities. In many situations it may be desirable to be able to verify the correctness of a message and at the same time, for privacy reasons, not reveal the true identity of the party transmitting the message. Examples include transmission of warning and informational messages to other vehicles and when exchanging information with road-side units.

- *Authentication and integrity check of external messages* exchanged between ECUs and external systems such as diagnostics equipment in repair shops, external servers when performing remote software updates, when changing configurations of vehicles, etc.

- *Confidentiality protection* of both in-vehicle and external communication. Closed communication V2X-groups may for example require encryption to maintain message confidentiality and integrity.
- *Identification of remote traffic* to and from servers and services that the vehicle, the driver or a passengers is using. Traffic from third parties, for example requests from agencies tracking vehicles for automatic road toll payments, also need to be properly authenticated.
- *Authorization of persons* such as owners, drivers, passengers and service technicians interacting with the vehicle. Examples include to control who can update a vehicle's software, drive the vehicle and/or order new functionality to it. In addition, some services may be based on the driver's identity such as insurance coverage and payment of parking fees, and not on the owner of the vehicle.
- *Diagnostic devices* should also function in an offline environment given that the authorized workshop may not have a internet connection. Additionally, it is important to be able to revoke a diagnostic device in case of theft.

6.3.1 Involved Parties

Trust is a complicated matter in the vehicular domain. As previously described, different entities involved may have different views of trust and privacy. It also depends on what application is being used, as some applications can be more trusted than other.

We have identified many different entities that need to interact with the vehicle during its lifetime. All of them have some interest in the functionality and can affect the safety of the vehicle, but the entities do not necessarily trust each other and even have similar interests:

- *The manufacturer* of the vehicle. They will continue to offer services to the vehicle during its full lifetime, not only in repair shops. Critical software updates have to be applied more or less immediately, not two or three years after a vulnerability has been discovered and the vehicle eventually visits an authorized repair shop.
- *The owner* of the vehicle. He/she should have access to most, but not all, functions in the vehicle. It should also be possible to delegate privileges to other users to configure some settings in the vehicle. The owner may change over time and transfers of ownership must be handled.
- *The driver(s)* of the vehicle. Many services will in the future be based on who is driving the vehicle.
- *Passengers* and authentication of passengers to access various applications and (remote) services, for example from e-commerce servers. Some services may be available in the vehicle based on a passenger's identity, for example multimedia contents or navigational software.

- *Authorized technicians/repair shops* should have access to most of the vehicle's data, but not necessarily to private information related to the owner, driver or passengers. The technicians should not be able to use their identities in, for example commercial activities, nor to access personal information such as driver's behavior, GPS logs containing vehicle location history, or to install non-authorized software.
- *Third party application suppliers* may be granted access to certain parts of the vehicle network, but not to all. This may be a rather complex issue to solve, similar to privilege levels offered for example for Android devices.
- *Trusted authorities* who may need access to certain data, for example after an accident or a crash or in real-time to track vehicles for various reasons; road tolls, parking fees, etc.

6.3.2 Certificate-based Authentication

Certificates are proposed to be used in all V2X applications that require authentication. Certificates can be distributed in some, but not all, broadcast messages or when new vehicles are identified by the sending node (see below). Despite the complexity with generating and distributing certificates, it is currently the only investigated alternative for authentication of vehicles and road-side units.

Certificate Revocation

Even if the identities of vehicles and road-side units are verified using certificates, they do not guarantee the correctness of the device. A road-side unit may be manipulated to send false information even though its identity can be verified by the vehicles. To limit the problem, certification revocation lists (CRLs) must in some way be distributed and be available to all vehicles. However, such lists can be extremely large and must be kept updated. Researchers are therefore investigating ways to implement more efficient distribution of large revocation lists and methods to reduce their size. Whether they will be efficient enough to work in reality even in densely populated environments remains to be seen.

Certification revocation lists are problematic to deal with since many organizations in different countries need to be involved. It also requires vehicles to either check all certificates on-line or to download huge lists of revoked certificates and keys. To make the list of revoked certificates shorter, the lifetime of certificates could be made shorter. The drawback is that vehicles then would have to connect to the CA more frequently to get new certificates. In some countries it may not be a problem, in other it may be. Vehicles lacking proper keys will therefore not be able to participate in communications with other vehicles, but they can still receive information from its surroundings. CAs may also have the possibility to tell vehicles to erase keys from its tamper-proof memory if they believe keys are compromised or misused. CRLs may also be distributed locally when a malicious vehicle has been detected.

Implementation Issues for Certificates

ECUs have limited cryptographic capabilities and the cost for performing such operations can be substantial. Most researchers agree that conventional X.509v3 certificates are too large for efficient and fast communication and the certificates to be used should be a subset of this standard.

Instead of using RSA/DSA signatures, Elliptic Curve Cryptography (ECC) is likely to be used which is substantially faster, although it is not unreasonable to believe that vehicles still need dedicated hardware to be able to verify signatures in real-time. The advantage of using Elliptic Curve Digital Signature Algorithm (ECDSA) instead of RSA signatures is that the time for signing the messages is much faster compared to RSA, however the verification is slower than using RSA [49]. ETSI defines in ETSI TS 103 097 security messages and certificates for ITS. A brief description of SecuredMessage and the certificate can be taken from the section *Secure Communication with the ETSI Security Layer* below.

Vehicle beaconing may be used for some services such as broadcast of information to neighbours about a vehicle's location and speed, for example every 100 ms. On a crowded road, a vehicle may receive, and be required to verify hundreds of messages per second. And for each vehicle, the corresponding certificate is also needed in order to verify its origin. It has been suggested [50] that certificates are attached not to every beacon transmission, but to a fraction of them in order to minimize overhead. Nodes, e.g. road-side units may also delay certificate delivery in their messages until a new node is detected, a scheme that would work quite well in smaller networks, which is a situation that is likely to exist for some time until all vehicles are equipped with this technology.

There are also other situations and applications where checking the signature can be omitted. It may be informational messages that are not essential for the vehicle, or it can be situations where on-board sensors and other systems can verify whether the message is authentic.

Private keys must be properly protected, and a Hardware Security Module (HSM) is a proposed solution to be used that offers tamper-resistance and makes it hard to steal other vehicles' identities, see Chapter 8.1.4.

ETSI ITS Certificate Authority Hierarchy

As shown in Figure 6.3, ETSI has introduced a hierarchy of CAs that aim to provide security services such as authentication and authorization for cooperative ITS communications. The ETSI CA hierarchy enables ITS stations to prove their identity to use ITS services and yet stay anonymous. Each CA entity issues/revokes digital certificates for ITS entities.

The ETSI ITS Security Trust and Privacy Management document [51] specifies the role of each certificate authority as follows:

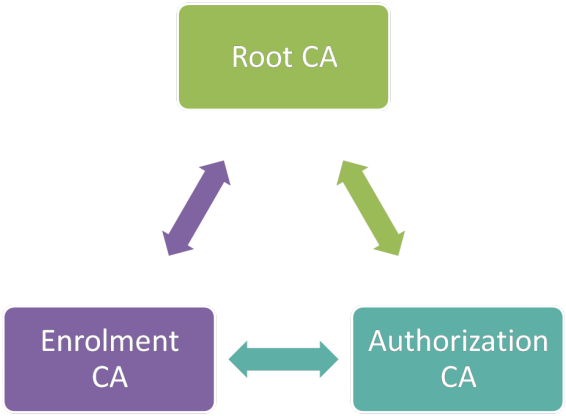

Figure 6.3: ETSI ITSC certificate hierarchy

Enrolment Authority (EA) issues Long Term Digital Certificate (LTC) to be used for authentication of the ITS stations. In order to participate in ITS communications and gain authorization tickets, it is necessary to have an LTC. To receive an LTC, first the ITS station uses its initialization credentials to request the enrolment credentials (long term certificate) from the EA. The process of gaining initialization credentials is performed in conjunction with the manufacturer of the vehicle or the ITS device [52]. According to the ETSI ITS [52] the initialization credentials are

- A canonical identity for the ITS station
- A public private key pair for the ITS station
- A generic profile of the properties of the ITS station
- A cryptographic certificate linking the canonical identity with its public key and generic profile

Next, the EA authenticates the sender of the request and issues a long term certificate for it. The issued LTC is then used by the Authorization Authority to authorize the ITS station [51].

Authorization Authority (AA) is responsible for granting specific permissions to the ITS stations. In order to authorize an ITS station, first it will be asked to provide its long term certificate to prove that it has been authenticated by an EA. Next, the AA checks the validity of the received LTC and finally issues authorization tickets (Pseudonym certificates). The pseudonym certificates shall be changed periodically according to the predefined time periods defined by the standard [51].

Root CA is the root of trust for all CAs and each ITS station should have access to at least one root certificate. A root certificate address might have been either installed by the manufacturer or broadcasted over the air [51].

Secure Communication with the ETSI Security Layer

ETSI defines in the ETSI TS 103 097 a security layer. This vertical layer is also shown in the ETSI reference architecture illustrated in Figure 6.2. The main security header described in [53] is encapsulated within the GeoNetworking stack and illustrated in Figure 6.4 [46]. The GeoNetworking protocol has been proposed by ETSI to provide Geo-Dissemination for vehicular communication.

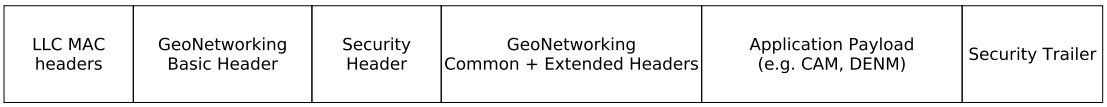

Figure 6.4: Structure of a secured Geonetworking packet (taken from [46]).

The structure of a secured message is shown in Listing 6.1. The *protocol_version* highlights the current version, version 2. The *header_fields* is of variable length and contains information required by the security layer, e.g. generation time, expiration, signer info. Each message can only contain one payload, sending multiple payloads in one message is thus not possible. The length of this field is variable and contains the messages like Cooperative Awareness Message (CAM) or Decentralized Environmental Notification Message (DENM). Furthermore, the *payload_fields* contains information about the payload type, such as unsecured, encrypted, signed_external, or signed_and_encrypted. The *trailer_fields* consists of information needed to verify the integrity and authenticity of a message using a signature. As the fields are going to be encoded or signed depending on the *security_profile*, each field has a numerical value assigned that specifies the encoding order (ascending) [46, 53].

Listing 6.1: Secured Message Structure (from [53])

```
struct{
        uint8           protocol_version;
        HeaderField     header_fields<var>;
        Payload         payload_fields<var>;
        TrailerField    trailer_fields<var>;
} Secured Message
```

ETSI TS 103 097 [53] defines the security profile for CAMs, DENMs, generic messages and certificates. CAMs are V2V messages sent by vehicles in a high frequency (10 Hz) to inform the surrounding vehicles about their position, velocity, acceleration, etc.. DENM are commonly sent by RSUs or forwarded by other traffic participants containing warning messages about slippery roads, traffic jams, or road works. Further, the standard recommends the use of *ecdsa_nistp256_with_sha256*, an ECDSA, for signing and verifying messages and certificates. The standard also defines that DENMs always need to contain the certificate, while the CAMs must contain the certificate_digest_with_sha256 element. Additionally, the certificate that is sent in the CAM messages should be sent periodically (1 Hz) or when a request_unrecognized_certificate message is received [46, 53].

The ETSI TS 103 097 [53] defines also the certificate structure for a V2X environment. The content of the certificate is shown in Listing 6.2. The *version* field contains the current version, version 2. *Signer_info* provides information about the signer of the certificate, such as self(0), certificate_digest_with_sha256 (1), certificate (2), certificate_chain (3), or certificate_digest_with_other_algorithm (4). The *subject_info* gives information about the subject name and type, for instance root_ca. validity restrictions can the taken from the field *subject_attributes*. The encoding for the signature is performed over all fields and in case a field has a variable length, the fields have to be ordered ascending according to their specified ID [46, 53].

Listing 6.2: Structure of an ETSI ITS certificate (from [53])

```
struct{
        uint8                   version;
        SignerInfo              signer_info;
        SubjectInfo             subect_info;
        SubjectAttribute        subject_attributes<var>;
        ValidityRestriction     validity_restrictions<var>;
        Signature               signature;
} Certificate
```

Hamida et al. perform in [46] also an experiment on an ARMv7 1 Ghz and a general-purpose 3 GHz Intel CPU. The results show that the system using the 3 GHz CPU is capable of performing 3831 signature generation operations per second and 817 signature verifications per second by using the ECDSA, ecdsa_nistp256_with_sha256, recommended by ETSI. The authors point out, that this performance might be enough for several tens of vehicles, but that the system would not be suitable for dense urban traffic (200 to 400 vehicles/km^2) without violating the critical-latency of around 100 ms for road safety applications. By being able to verify 817 signatures per seconds, one is able to verify messages from about 80 different vehicles and RSUs in case that the messages are sent in a frequency of 10 Hz. This rate does not seem to be sufficient, considering the radius in which the vehicle is able to communicate with other ITS stations and the fact that the vehicle receives many messages from stations driving the opposite direction [46].

6.3.3 Group Communication

Vehicles may also be participants of several VANETs at the same time, where groups are formed based on different properties. Group communication can, at least to some degree, solve the problem of network flooding when forwarding messages. By introducing group leaders, communication within the group can be more efficient, and group to group communication is done via group leaders only. However, the creation of groups is not trivial and algorithms for selection of group leaders are still being discussed. Traffic within a group should be signed with keys handed out by the group leader. When vehicles leave the group, keys can either be changed or the group can continue to live for some time.

Since IEEE 802.11p lacks security and does not support a Basic Service Set (BSS) mode like traditional WLANs using an access point, this has to be taken care of at higher

levels. The WAVE architecture addresses this issue and allows groups to be formed in a “software-implemented BSS mode” on higher levels. This allows closed groups to be formed where only participating members can exchange messages.

6.4 Threats to the Vehicle

With external communication, we mean all V2X connectivity as well as internet connectivity offered to vehicles. As described in Chapter 5, there are many protocols and potential weaknesses that need to be addressed. External traffic can be subject to traditional network attacks such as described by the Computer Emergency Response Team (CERT) at Carnegie Mellon University and many other sources.

6.4.1 Identity Theft

A vehicle’s or a person’s identity may be stolen. A Trojan planted in a vehicle may be used by a remote attacker to read, modify and even send data to a third entity pretending to be the (compromised) owner for example. It may be possible to use the victim’s identity in real-time transactions to sign messages for the benefit of the attacker. One example could be when a message about a road toll is sent to the attacker’s vehicle, it is immediately forwarded over the internet to a compromised vehicle to be signed, and then sent back to the attacker to be forwarded to the road-toll system. The consequences may be that the victim, i.e. the owner of the compromised vehicle is fined for traffic violations, parking tickets and road tolls. An attacker with access to many compromised vehicles may easily perform all kinds of Sybil attacks.

Information from earlier drivers may also be present in the vehicle and may potentially be reused or stolen. Different approaches have been suggested for how to maintain privacy of the vehicle and still be able to solve the problem of non-repudiation requirements. In short, all solutions conclude that keys used to sign messages should be short lived and changed regularly. This prevents stolen identities from being long-lived, although they do not solve the problem with real-time access to vehicles.

6.4.2 Access to the OBD-II Port

Since the internal networks lack protection, people and devices with access to the vehicle can perform all kinds of actions against the ECUs. The vehicle network is easily accessed through the standardized OBD-II port present in all vehicles. This port is used, for example by mechanics in repair shops to check vehicle configurations, change settings, update software and to read diagnostic error messages. The port interface is standardized and easy to connect to, although the internal messages are brand specific and some knowledge is needed to decode them. Some documented attacks have used reverse engineering approaches to figure out the meaning of these messages.

Computers, notebooks or even System on a Chip (SoC) devices can be connected to the OBD-II port which opens up for attacks “enhancing” the functionality in the vehicle. The port can be used by car owners who want to change some functionality, or by attackers

using an internet-connected PC as a gateway to the vehicular network. All traditional attack methods may be used by the attacker and must be addressed, for example in workshops and other environments where such devices are used. It is not unlikely to expect that many other third parties, for example official agencies performing yearly inspections of vehicles, will use this functionality to check configuration and diagnostic messages, and they all contribute to the complex problem of securing the vehicle.

There are devices available to be purchased such as the "*ELM327 Bluetooth OBD-II*", see Figure 6.5, that interfaces to the OBD-II port and offers Bluetooth connectivity to the CAN bus. The owner who installs the adapter can read, display and clear engine fault codes, view live engine sensors, etc. When installed, the safety of the vehicle depends on the driver's smartphone or PC and whether or not it is compromised. It may make the vehicle's network publicly available on the internet.

ECUs can also be reprogrammed through the OBD-II interface, although there is a security access code needed. The Unified Diagnostic Services Protocol (UDS) does not specify the length of the key nor the algorithm and thus it is a vehicle specific property. An evaluation of vehicle diagnostic security performed by Ring et al. in [54] shows that a 16 bit key is not sufficient, for the reason that it can be cracked with a brute-force attack, provided one has physical access to the interface, within approximately 110 hours. The protection with a 16 bit key is not very strong, although it prevents ECUs from being reprogrammed immediately by a malicious message. Increasing the key length or adding a delay between the authentication request and the response with a seed are two possible countermeasures.

Figure 6.5: OBD-II to Bluetooth adapter unit

6.4.3 Other external Communication Threats

In the vehicular domain, Jenkins and Mahmud [55] have discussed security problems and attacks against the vehicle and they look at both inter-vehicle and in-vehicle communications, and also at software and hardware attacks. Traditional methods derived from the CERT Taxonomy can be used, where we assume that the attacker can read, spoof, drop, modify, flood, steal, and replay traffic to the vehicle.

First, traditional attacks such as *DoS attacks*, like *buffer overflow attacks*, exhaustion of resources (memory, CPU computation power, network bandwidth) etc., must be addressed. Mechanisms similar to what is used in traditional computer-security (firewalls, MACs, encryption, etc.) can also be used to protect vehicles against external threats, at least to some extent. DoS attacks as well as jamming, broadcast tampering, and spamming have the availability of the wireless communication service as target. For instance, jamming attacks can be used to disturb the physical communication channel [46].

External network traffic can also be attacked or misused in ways that are more specific to vehicle communications. It may be possible to *replay* correctly signed transmissions for example at other locations, or to *modify* one's own messages and alter information such as position, speed and direction to get better service or cause confusion. Such attacks may be hard to detect since the messages can be properly signed by the vehicle sending out the information.

Certificates can be stolen/duplicated and illegitimate use of them may be hard to discover. Another vehicle's identity may be used, for example to obtain new firmware versions with enhanced functionality, to debit it for road tolls, parking fees, etc. Other attacks include impersonation such as identity theft and *Sybil attacks* where multiple identities are used for example to spread false congestion information.

Confidentiality is not a major issue for VANETs even though (signed) clear-text messages are broadcasted. Except for the identity of the vehicle, the information is in most cases intended to be public and to be used by whoever may feel affected. The exception is closed group communication where messages from some applications may use encryption. Clear-text communication has many advantages. It reduces cost of devices and may help to meet real-time requirements.

The real identities of vehicles should in most situations be considered as confidential. The use of pseudonyms may solve this problem, where a trusted third party can, if needed, identify the real vehicle behind the identity. If it is possible to always identify vehicles from their broadcast signatures, is it also possible to collect messages and vehicle identities, and by cooperating over the internet, to follow vehicle motions and publish their locations and movements in real-time on web sites. This is done today for ships through the AIS system at *marinetraffic.com*.

6.5 The SeVeCOM Project

The goals of the SeVeCOM project have been to create a component based security architecture that can have a very long lifetime and easily be extended and changed when new and unforeseen threats emerge [32]. They have concentrated security functionality into a *Security Manager* which is responsible for all security in all modules in the vehicle. It has hooks to the communication stacks (possibly created by different vendors) and can request to inspect and modify traffic at all layers. The security manager can implement services like firewalling, identity management and inspection of signatures in incoming messages. It is also the interface to a hardware security module (HSM or TPM)

which is physically protected (tamper resistant) and stores private keys and can perform cryptographic operations using these keys.

The idea with a separate security manager is that it is relatively easy to change functionality like cryptographic algorithms, keys, firewall and IDS functionality, in all communication stacks. Application programmers may, at least to some degree, therefore be relieved of considering all aspects of security and how to implement security services. It should, by using this kind of architecture, be possible to implement security into an existing network stack with minimal changes.

The *security manager* contains components for *identification*, *trust management* and *privacy management*. These components may subscribe to certain events from the hooks, for example to special types of messages.

The ideas presented are interesting and, as the authors point out, the use of hooks is similar to how Linux interfaces with its network stacks for similar tasks.

6.6 Remote Diagnostics and Software Download

Most of the work within securing remote diagnostics and software download has been directed towards the software download process and very little towards remote diagnostics.

Both unicast and multicast approaches have been proposed for secure remote software download. In [56], Mahmud, Shanker and Hossain describe a protocol by means of which software download is performed using an ITS infrastructure. The automotive company issues symmetric keys to encrypt the software transmitted between the software supplier and the vehicle. To increase the security in the transmission, they propose that the software should be sent twice and possibly also in random order to avoid attackers from predicting the message order. To authenticate the vehicle, a set of authentication keys are installed in the vehicle and also stored in a central server and transmitted to the appropriate Access Point (AP) within the ITS during authentication. The protocol was analysed in [57].

In the multicast approach proposed by Hossain and Mahmud [58], a special device denoted Network Device Monitor (NDM) is installed in the AP within an ITS infrastructure. The purpose of the NDM is to authenticate vehicles, manage the session keys for the multicast group, and to send software to the vehicles therein. A set of authentication keys are installed in the vehicle and also stored in a central server. These keys are transmitted and used by the NDM to authenticate the vehicle. Furthermore, digital certificates were used as authentication keys for authentication between the automotive company, the software supplier, and the NDM.

In [59], Nilsson and Larson propose a firmware update process where the firmware is split into smaller fragments and transmitted to the vehicle. Each fragment is hashed and the hash is concatenated to the previous fragment. Thus, all fragments needs to be hashed before any of them can be transmitted. The hash of the first fragment is used as an initial fragment containing a digital signature over the first hash, thereby

ensuring that all following hashes cannot be modified without detection. Encryption is also applied to the transmission. This protocol ensures data integrity, data authentication, data confidentiality, and data freshness.

Idrees et al. [60] give a detailed presentation of a remote software download procedure including some remote diagnostics, which utilises the HSM designed within the EVITA project.

Efforts are also made by ISO to create a standardized diagnostics protocol, Diagnostics over IP (DoIP) [61]. DoIP fulfils several use cases, such as reading ECU states, firmware upgrade and multi-purpose data transfers. This protocol is designed to use IP, which has the benefit of offering wireless and wired communication solutions. In [62], the authors have proposed a DoIP TCP protocol that allows performing remote diagnostic operations over the internet, for the reason that the DoIP vehicle announcement and discovery is depending on subnet broadcasts. The authors have performed initial tests, but did not implement a solution for firmware upgrades over the internet. However, appropriate security mechanisms are still missing in the DoIP-protocol.

A service used by Tesla to regularly update the vehicle's software is VMware AirWatch[1]. This product supports centralised management for multi-os deployments, such as TIZEN, QNX, Android, and Windows. ISO will extend its work in the diagnostics area by introducing security mechanisms in the electronic Periodical Technical Inspection (ePTI) standard to ensure data authenticity and integrity. The progress of this work will be further observed and evaluated during this project.

Finally, as the firmware has reached the ECU, reprogramming of the ECU needs to be performed securely. Methods for ensuring that the firmware is flashed correctly have also been proposed in [63, 64, 65].

To conclude, research in the area of secure remote diagnostics is still in progress. Solutions, such as the DoIP TCP in [62], look promising, but still need to be further investigated. Since there are great benefits of a remote diagnostics service, an architecture for secure remote diagnostics, including software download, should be defined.

[1] https://www.air-watch.com/ (2016-11-23)

Chapter 7

Internal Communication

In this Chapter we focus on the in-vehicle network, and architectures proposed to implement security into these networks are discussed. The CAN, LIN, MOST, and FlexRay protocols lack security functionality and were not designed with security in mind. All traffic is sent in clear-text and no protection against altered, spoofed or injected messages exists. When external communication is forwarded to these networks, appropriate security mechanisms must be in place. Border protection of vehicular networks can, and should, be done by the communication unit (CU) that handles external communication, but internal protection mechanisms must be present as well. Ideally, each ECU should be able to protect itself against malicious traffic and be able to identify the origin of all messages. However, there is a long way to go before these goals are fulfilled.

7.1 Communication Technologies

Vehicular systems use a number of different communication technologies with very specific characteristics. In the following, we discuss the bus technologies that are most prevalent in current automotive systems.

In-vehicle networks usually consist of several domains connected through a backbone, and each domain has one or more busses for different purposes. The most prevalent busses are CAN, LIN, MOST and FlexRay. There are also new technologies which are making their way into automotive systems: Ethernet has already seen its debut in the industry and is currently being standardized for use in automotive systems, and Controller Area Network flexible data-rate (CAN-FD) is also a prime candidate for future adoption. Each communication technology has particular speed and timing characteristics which makes it well-suited for a specific domain.

7.1.1 CAN

The Controller Area Network bus is the most common automotive bus: it is well established and well understood, and provides sufficient performance for most applications. CAN is a Carrier Sense Multiple Access/Collision Detection bus with Arbitration on Mes-

sage Priority (CSMA/CD+AMP). A node on a CAN bus can only transmit when it senses the transmission channel to be free. When two or more CAN frames are transmitted simultaneously, *bit arbitration* takes effect. As a result, the highest priority frame will be transmitted successfully while all other frames will be dropped. Thanks to this prioritization, CAN can be analyzed for its worst-case real-time properties, which is important for safety-critical systems. The typical speed of a CAN bus is 500 Kbit/s, with a *maximum speed of 1 Mbit/s*. A single CAN frame can carry a *maximum of 8 data bytes*. CAN busses are used for standard operational functions, but also for many safety-critical functions. In many vehicles, CAN is still used for the backbone network, although new backbones are gradually being implemented using FlexRay or Ethernet busses. A typical CAN frame is depicted in Figure 7.1. The arbitration field consists of a 11 bit ID, which is used for prioritizing the messages in decreasing order and to filter the messages. The data field has a variable length between 0 and 8 Bytes, where the length of the data field is transmitted in the control field. The Cyclic Redundancy Check (CRC) field is used to detect errors during a transmission. A valid CAN packet is confirmed with the acknowledgement field [66].

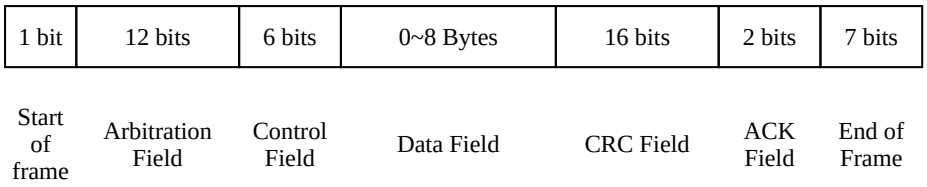

Figure 7.1: CAN frame format

7.1.2 LIN

The Local Interconnect Network bus was invented as a cheaper alternative to the CAN bus. LIN follows a star-topology with one master node and at least one slave node. Communication on the bus is controlled exclusively by the master node, so Collision Detection (CD) is unnecessary, and latencies are guaranteed. The master can itself acts as a slave. The maximum speed of the LIN bus is 19.2 Kbit/s, and a single frame can carry a maximum of 8 data bytes. It is mostly used to control vehicle body electronics, such as windows or windshield wipers.

7.1.3 MOST

MOST[1] is a network that is specially designed for transmitting multimedia and infotainment content. This synchronous network is realized with a ring topology and offers a total bandwidth of approximately 23 MBaud, which is divided into 60 channels. The maximum number of nodes in a MOST network is 64. Synchronization is achieved

[1] http://www.mostcooperation.com/technology/most-network/ (2016-11-22)

with a *TimingMaster* that transmits the system clock with a continuous signal. All other devices/nodes are named *TimingSlaves*. The network is synchronous, but it allows both, asynchronous and synchronous data transfer. Moreover, MOST defines all layers of the OSI Reference Model. Advantages of MOST are its efficiency in transmitting multimedia content and that it offers electrical and optical solutions [67].

7.1.4 FlexRay

FlexRay was designed to be faster and more predictable than CAN. It achieves the higher predictability by using Time Division Multiple Access (TDMA) instead of Carrier Sense Multiple Access (CSMA). FlexRay works in cycles, and there are two main types of cycle segments: static and dynamic. Static segments are divided into a number of fixed-length slots, each with a specific ID, and only nodes with the corresponding ID are allowed to transmit in that time slot. This allows predictable communication and supports applications with hard real-time requirements. Dynamic segments allow for event-triggered communication by following the same time slot arrangement as static segments, but if a node needs to transmit more data than fits into its time slot, it can "take over" time slots of nodes with lower IDs, forcing the nodes with lower IDs to wait for a free slot. In order to work correctly, FlexRay requires synchronized clocks, and clock synchronization is built into the protocol to guarantee the required accuracy. The maximum bus speed is 10 Mbit/s, and a single frame can carry a maximum of 254 bytes of data. FlexRay allows for a fault-tolerant dual channel approach, and supports different types of topologies. Its main drawbacks are that it is more expensive and more complex than CAN.

7.1.5 CAN-FD

CAN's bandwidth limitations have given rise to a new CAN-based technology called Controller Area Network flexible data-rate, which is also backward compatible. As the name suggests, CAN-FD is CAN with a flexible and higher data rate: CAN-FD has a *maximum speed of 2 - 5 MBit/s*, depending on the bus topology. A single CAN-FD frame can carry a *maximum of 64 data bytes*. It achieves its backward compatibility with CAN by differentiating between the arbitration phase and the data transmission phase. During the arbitration phase, it uses the same bit rate as classical CAN, but during the data phase CAN-FD can switch to higher bit rates. The stand out advantage of CAN-FD over FlexRay or Ethernet is that it is backward compatible to CAN and thus cheaper to implement.

7.1.6 SAE J1709

The SAE J1709 allows the serial communication between ECUs and is specially designed for the use in heavy duty vehicles. This standard recommends twisted pair cables to achieve a bandwidth is 9600 bits per second. Simplicity and low cost were the key benefits for using this standard. Messages consist of three fields: the *Message Identification Character (MID)*, *Data Characters*, and the *Checksum*. The MID ranges from 0 to 255, where 0 to 68 is defined by the standard, 69 to 111 and 128 to 255 is assigned by other committees, and 112 to 127 is available for general use [68].

7.1.7 SAE J1939

The SAE J1939 [69] is a common communication architecture that has been defined by SAE due to the need of a communication standard between ECUs from different manufacturers for heavy duty vehicles. It defines the message timeouts, the network speed, and vehicle body control. The physical layer in SAE J1939 is the CAN bus. Junger lists in [70] the following as particular characteristics of J1939:

- Extended CAN identifier (29 bit)
- Bit rate 250 kbit/s
- Peer-to-peer and broadcast communication
- Transport protocols for up to 1785 data types
- Network management
- Definition of parameter groups for commercial vehicles and others
- Manufacturer specific parameter groups are supported
- Diagnostics features

SAE J1939 splits the CAN identifier into several fields that indicate among others the priority, parameter group number, and source address. Parameter group numbers (PGNs) define the content of the messages. For example, PGN 65262 *Engine Temperature* contains 8 Bytes of data representing the Engine Coolant Temperature, Fuel Temperature, Engine Oil Temperature, Turbocharger Oil Temperature, Engine Intercooler Temperature, and the Engine Intercooler Thermostat Opening. Additionally to the defined content of this PGN, J1939 also defines the transmission rate which is in this example 1 second [69, 71].

For the reason that this standard defines the structure of the transmitted frames, it is not possible to add security measures, such as Message Authentication Codes, in the same frame. A possible method can be sending this information as an additional frame for specific data frames belonging to the general use group, which has the disadvantage that the ECU would have to wait until the second frame arrives.

7.1.8 Automotive Ethernet

Ethernet can be the solution for the need of a higher bandwidth. It causes a paradigm shift in the design of the vehicle's internal network, for instance subnets can be used for segmentation. As Hank et al. state in their paper [72], the TCP/IP stack is already in use when charging the vehicle to allow a communication with the vehicle's charge control module. This charging process is defined by the ISO 15118. Moreover, DoIP (briefly described in Section 6.6) takes also advantage of the existing IP stack.

Important aspects of using Ethernet in the vehicle are the costs for the wiring and the chipset, the bandwidth, and the Electromagnetic compatibility (EMC). The electromagnetic emission profile is critical for the reason that the FM radio band must not be disturbed by cabling. The BroadR-Reach[2] technology offers a bi-directional communication that operates on an unshielded twisted-pair cable, has a symbol rate of up to 66.6 MBaud and a bandwidth of 100 Mbit/s [72].

7.1.9 Audio Video Bridging (AVB)

Audio Video Bridging (AVB) is designed for the standard ethernet network technology and has the benefit of providing well synchronised streams in realtime by using a simple cabling. AVB is a solution for the need of synchronised video and audio streams due to the increased presence of camera devices to provide a surround view. This technology is also intend for the use as a backbone for control data. The real-time requirements are achieved with a time synchronization and quality of service.

The following IEEE specifications are part of the IEEE AVB (taken from [73]):

- **IEEE 802.1BA [74]** - Audio Video Bridging Systems Overview
  Introduction and Overview
- **IEEE 802.1AS [75]** - Timing and Synchronization for Time-Sensitive Applications
  Time synchronization of the nodes by using a reference time with an accuracy that is better than 1 micro second.
- **IEEE 802.1Qav [76]** - Forwarding and Queueing Enhancements for Time-Sensitive Streams (FQTSS)
  FQTSS - Traffic Shaping, separation of time critical and not time critical traffic.
- **IEEE 802.1Qat [77]** - Stream reservation Protocol
  Reservation of resources done by using buffers and queues within the bridges.

Jesse concludes in his presentation [78] that AVB can be used in parallel with IP since the frames are already separated on driver level and this ensures the high availability of AVB. This technology is also designed to allow a maximum hardware support, e.g. use of FPGAs [78].

7.1.10 Time-Triggered Ethernet (TTE)

The TTTech Computertechnik AG[3] offers a solution called Time-Triggered Ethernet (TTE). This technology is based on Ethernet and provides a "*fully deterministic communication, fault-tolerant synchronisation services, guaranteed constant latency for multi-hop communication routes in the network, and partitioning/protection of network traffic*" [79, p.1].

[2] https://www.broadcom.com/application/connected_car.php (2016-11-29)
[3] https://www.tttech.com/ (2016-12-16)

TTE categorizes traffic in three different groups: time-triggered (TT), rate-constrained (RC) traffic, and best-effort (BE) traffic. The sender and the TTE switches have a transmit or forward schedule, when sending TT traffic. This traffic class requires a clock synchronization among all devices. RC traffic is ensured to provide lossless communication with a bounded latency and jitter. Moreover, the sending node gets a reserved bandwidth. The BE traffic class is compatible with the IEEE 802.3 standard and provides no timing guarantees [79].

7.2 Threats to the internal Network

Most work with security has focused on identifying and showing the lack of security in vehicular systems and the need for protection mechanisms, rather than specifying how the problems should be solved given the special requirements and restrictions that vehicular communication has.

Many researchers have shown that there is a significant lack of security mechanisms in in-vehicle networks. Koscher et al. [1] conducted experiments on two vehicles and by using techniques such as packet sniffing, packet fuzzing, and reverse-engineering, they found a number of attacks that could be performed against the in-vehicle network. Wolf et al. [80] did some early investigations of possible attacks against different busses in the in-vehicle network and demonstrated the lack of security. The results from these studies may not be very surprising since security is not designed into the protocols being used (CAN, LIN, MOST) and any device, an internal ECU or an external device, that is able to send messages can forge arbitrary legal messages.

Hoppe and Dittmann [81] used simulations for evaluating security. They investigated the possibility of performing sniffing and replay attacks on the CAN-bus by simulation of an electronic window lift system. To classify their attacks, an adapted version of the CERT Taxonomy proposed by [82] was used. This taxonomy classifies the vulnerabilities into three classes: *design, implementation* and *configuration*. In a later study, they performed attacks against the electronic window system using real hardware as well as attacks against the warning lights of the anti-theft system and the air-bag control system [83].

Nilsson and Larson [84] introduced the concept of a vehicle virus. The virus was listening for the message on the CAN-bus that locks the doors remotely, and when that message was captured, the virus would soon afterwards unlock the doors and start the engine.

There are several types of security problems that have been, and still can be, exploited that must be addressed:

- *lack of sufficient bus protection.* The CAN-bus lacks necessary protection to ensure confidentiality, integrity, availability, message authenticity, and non-repudiation [83]. Messages on the CAN-bus can be read by other nodes, they have no sender or receiver address (only message types that ECUs can subscribe to), and are not

protected by any MAC or digital signature and lack necessary protection of data authentication, data confidentiality and data freshness.

- *weak authentication.* It is possible to illicitly reprogram ECUs with new firmware [1]. The reason is weak authentication and sometimes no authentication at all, see Section 6.4.2.
- *misuse of protocols.* Attacks against the in-vehicle network can be performed by misusing well-chosen mechanisms in the protocols [80]. On the LIN-bus, sending malicious sleep frames could disable the whole network, and on the CAN bus, a DoS attack may be carried out by misusing the bus arbitration mechanism by continuously sending messages with the highest priority, resulting in that no one else can access the bus. Furthermore, well-formed malicious error messages can be used to attack the fault detection mechanism implemented in CAN and FlexRay and cause ECUs to disconnect from the network.
- *poor protocol implementation.* In some cases the protocol implementation is such that it does not properly reflect the protocol standard. For example, for safety reasons the standard specifies that it should not be possible to put the vehicle or its ECUs into programming mode while the vehicle is moving. However, in some implementations it is indeed possible to launch a command that would disable the CAN communication and put ECUs into programming mode despite the fact that the vehicle is moving [1].
- *information leakage.* Information leakage from the vehicle can be triggered by manipulating the diagnostic protocol, creating a potential privacy violation. Hoppe et al. [85] have demonstrated this by sniffing an ordinary diagnostic session, and then replayed it with slightly modified commands. Since the gateway ECUs are unable to distinguish between ordinary traffic and diagnostics traffic, both types of traffic will be forwarded and processed.

7.3 Need for a planned Architecture

By having a proper internal architecture similar to what is proposed by the Evita project [4], some threats can be eliminated or at least the consequences, from a security point, can be limited to the subnet (bus) from where it originated, see Figure 7.2. There are several attempts to deal with this problem, but the main constraint that limits the applicability of solutions is the cost of the solutions, especially if it requires more powerful ECUs. Without these limitations, conventional security mechanisms used to protect corporate networks from malicious internet traffic could be used directly.

A defense-in-depth approach for securing the vehicle is discussed by Larson and Nilsson [86]. They discuss how to prevent unauthorized access, use IDSs and logging mechanisms for detection, IPSs as a countermeasure, use honeypots for information retrieval and detection of new attack methods, and the necessity of traceability to perform recovery. In [87], Nilsson and Larson extend this discussion.

A well-planned architecture with traffic separation, internal firewall functionality preferably in all ECUs with possibility to detect malformed, spoofed and incorrect messages is the overall goal. This can be accomplished in different ways. Some approaches take just a few steps in this direction, other try to solve the overall problem but with higher complexity and cost as a result. In the following paragraphs we investigate some proposed solutions.

7.3.1 Leave Access Control to higher Layers

Chavez et al. suggest the use of the security services of the OSI Reference Model (ISO 7498-2) for securing the CAN-protocol [88]. The OSI model describes five security services, confidentiality, integrity, authentication, non-repudiation, and access control. According to this, they propose that access control should be taken care of at higher layers in the protocol, that integrity should be enforced by using hash algorithms, and that confidentiality should be enforced by using RC4 encryption of the CAN-frames. The remaining two OSI services, authentication and non-repudiation were not considered to be useful in this context. Apart from resource constraints with encrypting all messages, key distribution between internal nodes is a major problem.

7.3.2 The EVITA Use Case Architecture

The EVITA project has designed a reference architecture that is useful when discussing vehicular networks. The vehicle network is divided into sub-networks controlled by Gateway ECUs. These gateways can have some firewall functionality built-in to protect their network from unwanted communications to and from the other networks. They can make sure that most messages on the local network remain local and that only selected messages are sent to other networks. Similarly, they can restrict what messages may be forwarded to the local network.

External communication is mainly done through the Communication unit (CU) but with some exceptions (as shown in Figure 7.2): USB and Bluetooth communication is performed by the multimedia subsystem.

This model is not entirely new. Car manufacturers (e.g. Volvo, BMW and Volkswagen) already use multiple internal busses for separation of traffic, although they use their own proprietary designs. If such a standard will emerge or whether each manufacturer will use their own, remains to be seen. The model is still very useful and can be used in discussions of separation, firewalls, IDS functionality, etc., and it is used as a basis for discussion in this deliverable.

7.4 Possible Security Mechanisms

To secure internal communications, several traditional mechanisms have also been proposed. These include message authentication codes (MAC) for traffic integrity, firewalls both for external traffic and for internal traffic implemented in gateway ECUs, use of IDSs to detect unusual activities on the networks, certificates for identification of various

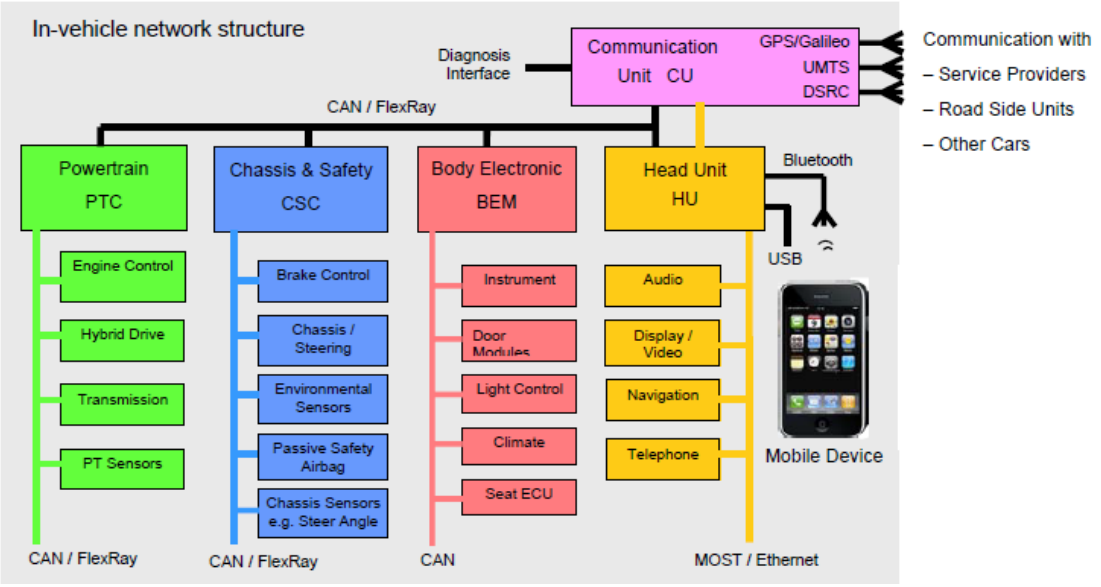

Figure 7.2: Evita project use-case architecture

devices (vehicles, road-side objects, drivers and ECUs). These and many other mechanisms are described in the following text. Figure 7.3 provides a summary of papers describing protection mechanisms in different areas.

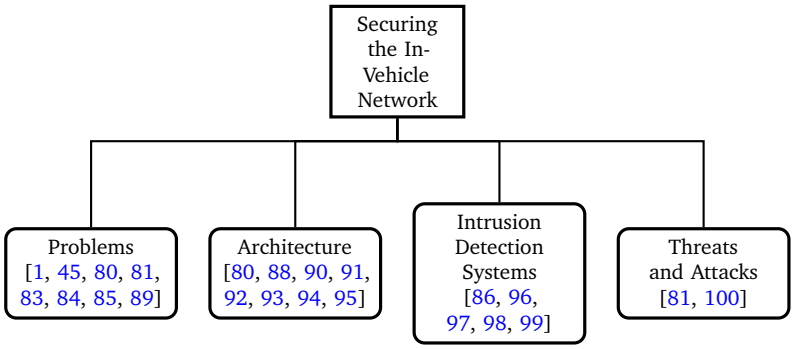

Figure 7.3: Protection mechanisms for in-vehicle networks

Message Authentication Codes (MACs) can be used to provide integrity of the messages. However, since internal security is not standardized, this means that in order to implement MACs, it is necessary to modify the existing protocols (see Section 7.4.2)! Other approaches that have been proposed is to create new security architectures, for example with gateways limiting cross-traffic between different parts of the vehicle. However, these approaches still have to be evaluated considering the limited resources of the in-vehicle network. In addition, they do not explain to application writers how security should be

implemented, what traffic needs to be filtered in ECUs and gateway nodes, and what subnets should be implemented in their particular vehicles and models.

Firewalls can be used to protect both in-vehicular traffic and traffic from external sources. For example, Wolf et al. [80] briefly discuss the concept of an internal firewall in each ECU, but we know of no attempts to really introduce a firewall where traffic is filtered by ECUs. We also note that out of the four protocols used for the in-vehicle network (CAN, LIN, MOST, and FlexRay), almost all research have addressed CAN and very little work with the other protocols have been done.

Intrusion Detection Systems have been studied to some degree, and both anomaly-based and specification-based IDS have been suggested for the CAN protocol. However, no approaches have been found for other protocols, and since FlexRay also lacks appropriate security mechanisms and eventually will replace the CAN-protocol, an IDS for FlexRay should also be investigated. A longer description of IDS systems can be found in Section 7.5.

7.4.1 Trusted Communication Groups

Groll and Ruland [91] propose an architecture where they divide the in-vehicle communication into trusted groups. All ECUs within a trusted group share the same symmetric key to encrypt and decrypt the communication. A Key Distribution Center (KDC) within the vehicle is used to create and distribute the symmetric keys for these trusted groups. The trusted groups are defined by access control lists, ACLs, and are signed by the automotive company. One ACL is stored in each ECU and defines the trusted groups that the ECU belongs to.

To distribute the symmetric keys for communication within the trusted groups to an ECU, the ECU sends its ACL to a Key Distribution Center, KDC. After the KDC has verified the signature on the ACL, the KDC sends back the symmetric keys for those trusted groups defined by the ACL. To protect the distribution of the trusted group keys, asymmetric encryption is used between the ECU and the KDC. The asymmetric keys needed must also be signed by the automotive company.

A key exchange protocol based on EVITA's HSM to establish trusted group communication between ECUs are proposed by Schweppe et al. [95]. By using two symmetric keys and tag them with special flags the keys can be used for encryption and signing comparable to an asymmetric encryption approach. Keys are managed by a key master and a detailed protocol on how keys are distributed between the involved ECUs are given. Encrypted messages can then be transmitted over CAN using the enhanced common tranport protocol (CTP) based on ISO 15765-2. Both direct and group encrypted communication are supported.

An extended version of HSM functionality has been suggested by Oguma et al. [92]. They propose an architecture where only ECUs with successfully validated software will be able to exchange symmetric keys for further encrypted communication. It uses three component types:

- a center outside the vehicle,
- a master ECU within the vehicle, and
- all other ECUs within the vehicle.

The center stores the information about all vehicles, and the master ECU in the vehicle is used to do local validation since the center might not always be reachable. The master ECU holds a list of hash values that are valid for the software running on each ECU in the vehicle. Instead of using asymmetric encryption within the vehicle, a Key Predistribution System, KPS, is used. After the validation process has been performed and the software in the ECUs are authentic, keys are generated in the KPS for each pair of validated ECUs, keys to be used to encrypt and/or sign messages. When the keys are distributed, the messages also contain information which can prove that the ECU has been validated and, among other things, a counter to protect against replay attacks. Lee et al. [101] further discuss the attestation-based security architecture. By using ProVerif, they propose a way to formally verify the protocol used to distribute these keys.

A different approach to the problem has been suggested by [94]. This is accomplished by the introduction of a data management system, DMS. Instead of letting all ECUs exchange data with each other, data is stored in specific nodes within the vehicle. By using a DMS for storing data, security mechanisms such as access control could be enforced on access of data and to ensure data integrity. The method also opens for the possibility to store a global state to a protective storage in the case of an accident. Three different approaches to deploying the DMS were investigated: a centralized approach, a distributed approach, and a hybrid approach, in which a DMS is deployed for each sub-network. The hybrid DMS approach was found to be the most attractive.

7.4.2 Authentication of ECUs

Wolf et al. [80] suggest ways to improve the security of the communication by requiring authentication of the ECUs and by encrypting the communication. First, each ECU has to be authenticated by the gateway by means of a certificate. After authentication, the ECU will receive a symmetric encryption key that is shared with other authenticated ECUs on that local network to make secret data exchange possible. When using this method, it is inevitable to secure the gateway against unsigned firmware updates and other attacks, as a compromised gateway would lead to an entirely compromised system.

7.4.3 Authentication of multiple Destinations

An approach to provide authentication of messages to multiple destinations for time-triggered applications is proposed by Szilagyi and Koopman [93]. A protocol was designed to be able to authenticate multiple destinations at the same time, which requires that each pair of communicating nodes share a symmetric encryption key. These keys are used for calculating the MAC over the messages for each destination. Each MAC is further stripped down to a few bits and concatenated to the end of the message. Since it is easier

to forge a message with only a few bits of the MAC available, the authors propose that authentication is provided by successfully verifying the MAC over a set of messages. For the two types of messages investigated, state-changing messages and reactive control messages, an upper boundary of the probability of performing a successful attack is discussed.

The proposed protocol also has protection against replay attacks. The protocol is further discussed in [102], where an analysis with the help of simulated attacks is provided.

7.4.4 Message Authentication

Nilsson et al. [90] propose the use of a MAC for providing data integrity and data authentication in the CAN communication. To achieve this, a 128 bit key is shared between the two communicating ECUs. The MAC is calculated over four consecutive CAN-messages and the resulting MAC is divided into four 16 bit blocks and transmitted in the CRC-field of the next four CAN-messages. The protocol introduces a delay before the data integrity and data authentication can be verified. In total, eight messages are needed for the verification to be completed. Two of the remaining challenges with the protocol were that if the MAC calculation fails, the actual individual message that was wrong can not be identified, and that there is no protection against replay attacks.

7.5 Intrusion Detection and Prevention Systems

IDSs has been suggested for all types of systems and environments, including inside vehicles, for some time but have for various reasons not yet really been that successful. The main problem is how to handle both false positives (false alarms) and false negatives (failure to detect a problem). There is usually a strong relationship between the two: a high detection rate also results in a high rate of false alarms. In the vehicular domain, it is not clear what should be done when a potential problem is detected. False alarms may cause more problems than the problem itself would have done. Both specification-based and anomaly-based detection methods exist, as described in the following paragraphs.

Kang et al. provide in [66] a novel approach for an IDS using a Deep Neural Network (DNN) to secure the in-vehicle network. The learning was achieved with an unsupervised pre-training method and following stochastic gradient descent method. The authors evaluated their proposed technique within a simulation environment (OCTANE [103]). The results show an accuracy of 98% on average [66].

7.5.1 Specification-based Detection

Larson et al. [86] propose and evaluate a specification-based IDS for the CAN 2.0 and CANopen 3.01 protocols. They conclude that, since these protocols lack information about the producer and consumer of messages, there is not enough information available for using network-based intrusion detection. Instead, they propose host-based detection, i.e. one detector is placed in each ECU. Incoming and outgoing traffic can then be

investigated based on information from the protocol stack and the object directory of the CAN-protocol at the specific ECU. For the detector in the ECU, security specifications for the communication protocol and the ECU behavior can be developed.

The authors conclude that the gateway ECU is the most important ECU to protect. If the gateway ECU is compromised, all types of attacks can be performed. Unfortunately, performing detection in the gateway ECU is harder than in ordinary ECUs since the detectors for the different interfaces at the gateway have to cooperate to detect certain attacks, e.g. to detect lost or modified messages.

7.5.2 Anomaly-based Detection

Anomaly-based detection has been studied in ordinary desktop environments, but a problem faced in this environment is how to define a normal network behaviour; The desktop network is provided for arbitrary network communication. In contrast, for in-vehicle networks researchers argue that the expected payload is more well defined as the messages transmitted in the is specified during the design and anomaly-based detection should be easier to implement.

Hoppe et al. [97] demonstrate an anomaly-based IDS for the CAN protocol. In contrast to the specification-based approach by Larson et al. [86], where the IDS is placed in the ECU, they listen to the network traffic on the CAN-bus. By looking at the rate of how often specific messages are transmitted on the bus, and comparing that to what is considered to be normal, deviations from the expected number of transmitted messages can be detected. This was exemplified by investigating the system that detects physical vehicle break-ins. When the anti-theft alarm is activated, the system sends messages to the lights of the vehicle to turn them on and off, so that they are flashing. An attacker does not want these lights to be activated, but since the CAN-bus is a broadcast network, messages sent by the alarm system can not be deleted (except possibly in gateways). Instead the attacker have to create new messages to turn the light off as soon as it is lit. These new messages will be a deviation from the normal messages sent, and detected by the anomaly-based IDS.

Another method proposed by Hoppe et al. in [104] is the identification of misuse of messages. ECUs send their messages with a message ID on the CAN bus. Thus, the sending ECU is able to check if messages with its exclusive message IDs are sent from another source on the same CAN bus. This approach can be applied to gateway ECUs as well, as they know from which subnet which message IDs are sent. Matsumoto et al. present in [105] a similar approach with the extension, that the ECU detecting anomaly is sending an error frame immediately in order to override the malicious frame.

Core building blocks for implementing an anomaly-based in-vehicle IDS are described by Müter, Groll and Freiling [98]. They define eight types of sensor to detect different anomalies in the in-vehicle network and six criteria on sensor operations. For each of the criterion they evaluate how the different sensors will perform. The eight sensors are defined with respect to:

1. *formality* and correctness of the messages transmitted, i.e., the correctness of packet headers, checksums, etc.,
2. the *location* of the message, i.e., in which sub-network the message is transmitted,
3. the *range* of values in the payload of the message, i.e., within what boundaries a value need to be in,
4. the *frequency* of the given message,
5. *correlation* of message occurrence between different sub-networks,
6. different aspects of the inspected *protocol*, e.g., challenge-response behaviour,
7. *plausibility* of the transmitted payload, e.g., vehicle speed is increased instantly from 0 km/h to 100 km/h, and
8. the *consistency* of the over all system, by help of other sources in the vehicle.

The six criteria describes how the different sensors can detect anomalies: (1) based only on deviation from the message specification such as given by the CAN-Matrix, (2) the number of messages needed to detect an anomaly, (3) the number of sub-networks needed to perform detection, (4) how many different types of messages are needed to identify the anomaly, (5) wherever the payload needs to be inspected, and (6) wherever the semantics of the message needs to be considered. A classification of the sensors are also presented, based on how many messages are needed for detection and whether the payload needs to be inspected. Finally, they present a model of how the severity of the sensor alerts are mapped to the three levels of alert (to get the driver's attention) proposed by Hoppe, Kiltz and Dittmann [96].

An information-theory approach by analysing the entropy of transmitted messages are proposed by [99]. They were able to detect a set of anomalies in their test-setup by identifying a deviation in the entropy of the transmitted messages. Among these were the detection of increased frequency of messages and message flooding.

The SeVeCOM project [32] recommends vehicles to use an anomaly-based IDS system internally. However, they do not address in detail how and what the IDS system should do except that "appropriate reactions should be taken to get the system back to a secure and safe state". As we see from the proposals above, some approaches have been suggested.

Ling and Feng propose in [106] an IDS that is listening to the CAN messages, and creates and updates a table of counters that indicate whether the message ID is from a known input set or unknown to the system. This approach requires the specification of all known message IDs that are sent via CAN and thresholds for known and unknown messages. With the use of flags and the threshold values, the system performs an alarm operation in case the counter exceeds the threshold. The algorithm was evaluated in a simulation environment using CANoe [106].

Cho and Shin propose in [107] an anomaly-based IDS called Clock-based IDS (CIDS) that measures the intervals of periodic in-vehicle messages and creates a fingerprint of the ECU. With this method, one is able to distinguish between several source ECUs due to each ECU's individual offset and skew. Thus, this approach is able to identify the ECU that is sending the malicious messages.

7.5.3 Handling Intrusion Alerts

One crucial issue with intrusion detection is to decide what to do with an alarm. It may be possible to send the alarm to a central portal where a security officer takes care of it. However, it may not be realistic to assume that the portal should have such resources for the large number of cars connected. Further, the car may not be continuously connected to the portal. Thus, it seems more realistic to inform the driver of the alarms. Such an approach is proposed by Hoppe et al. [96] where various security-related events can be presented to the driver. Depending on the severity of the event, different methods are used: visual for non-critical events, acoustic for critical events, and haptic for severe events.

They also propose an "adaptive dynamic decision model". By using the sensors in the vehicle, the environment of the vehicle can be evaluated at the time of the alert. If the currently used ways of alerting the driver is not considered to be enough, the alert-level can be increased.

Since the communication within the vehicle is safety-critical, discarding the wrong message may have catastrophic effects. An IPS system is essentially an IDS system that can take some action, for example to stop a certain behavior. However, an attacker use the fact that an IPS system is present and force it to make incorrect decisions. Hoppe et al. discuss the problem of intrusion response and point out that an active response system might not be allowed to actively make decisions in the vehicle due to legal requirements for safety-critical systems [97].

7.5.4 Honeypots

A honeypot is a faked target that is intended to attract attacks in order to collect information and to analyze the attacks. To our knowledge, only one such approach has been described so far, by Verendel et al. [108]. It is suggested that the honeypot is attached to the gateway node in the vehicle and simulates the in-vehicle network. The data collected from the honeypot can then be sent to a common portal and analyzed in detail. The purpose of this is to learn about new attacks and distribute solutions as early as possible.

A very important property of the honeypot is how realistic the simulation of the target is. If the simulation is not realistic enough, the attacker will realize that he is not attacking a real vehicle. However, making a realistic honeypot may be very hard since it is not likely that a real vehicle is used for this purpose.

Chapter 8

Hardware and OS-level Security

This Chapter contains a description of available security mechanisms for hardware and operating systems and examines how they can be used in the automotive domain.

8.1 Hardware Security

Hardware for embedded systems traditionally deals with many more restrictions than general-purpose hardware. Therefore, advances in embedded hardware are often the result of reducing power consumption or costs of already known solutions. Since even in general-purpose hardware has been a shift towards more power efficient solutions, this trend will probably continue or accelerate.

There are a number of issues that are interesting with regard to hardware security, such as supported security features, known attacks that need to be mitigated and the current state-of-the-art in automotive microcontrollers. We will discuss these issues in the following Subsections.

8.1.1 Security related Microcontroller Features

As shortly discussed in Section 2.3, there are a number of security features that are supported by modern CPUs and microcontrollers.

One very obvious feature are hardware implemented cryptographic modules, e.g. hardware support for common algorithms such as AES, RSA or SHA-1, which can greatly speed up encryption or decryption routines.

Moreover, many microcontrollers can distinguish between executable and non-executable memory spaces, which makes certain kinds of attacks harder to perform (e.g. the traditional buffer overflow attack).

Virtualization known from the server and computer environment has moved to embedded systems as well. However, embedded hypervisors have compared to traditional hypervisors different attributes and requirements. Efficient memory management and usage is important in the area of embedded systems, as the memory resources are limited.

The memory limitation requires also small hypervisors, which have the advantages that their code is easier to validate and the code can be proven to be bugfree. In the automotive domain it is also highly important that the hypervisor provides a strong separation of the applications, since a safety critical application may be executed on the same SoC as the infotainment application. Furthermore, an embedded microcontroller must be capable of handling the real time requirements [109].

8.1.2 Side Channel Attacks

Side channel attacks are one of the main types of attacks that hardware needs to guard against. They are generally very time consuming and costly, and an attacker needs to be quite skilled to pull it off. Common side channel attacks are differential power analysis or differential electromagnetic analysis (DEMA) [110].

8.1.3 Tamper Detection Modules, TDMs

Tamper detection modules need to detect attempts to directly tamper with the hardware and trigger appropriate counter measures. For instance when the lid of an ECU is removed by unauthorized people, it may be appropriate to delete stored keys in order to prevent that they are stolen. However, it this is far from trivial.

8.1.4 Hardware Security Modules, HSMs

A hardware security module (or a Trusted platform module, TPM) contains security-critical functionality needed by other components of the vehicle, such as to protect private keys (used in asymmetric encryption), to distribute session keys, and to sign messages. It contains memory, a processor and software capable of performing basic cryptographic operations and preferably also a good random number generator. It should ideally be a tamper-proof device and be protected against physical access, i.e. that all attempts to access its content should make it useless.

A valid and reliable clock is needed by many services to avoid replays of messages and to detect reuse of older messages in other environments. The HSM module is a good place for such functionality and a timestamp can be applied to the message at the same time as it is digitally signed. This prevents individual ECUs which are compromised from sending and reusing old messages.

HSM functionality is implemented in special hardware similar to the well-known IBM 4578 cryptographic processor with tamper-resistant protection. It should have a well-defined API that can be used by ECUs to perform crypto-related operations such as to sign and verify signatures. Smart-cards have been proposed for some situations, but one problem with them is that they lack functionality such as offering a trusted time source. In situations where this is not needed, smart cards and RFID cards offering crypto-operations can be used, for example when identifying owners, drivers and service technicians.

HSMs have also been suggested to be used by internal ECUs in order to guarantee execution of authentic code. The functionality can be similar to Windows notebooks, where a TPM chip together with the first boot-loader (BIOS) verifies the integrity of the software before storing it and only authentic (genuine) software will be executed. This can be done in steps and even include signed applications from third party developers. All non-trusted and modified software, including malware, will be rejected by this system. There are many advantages with this solution, although the drawback is the increased complexity of the ECUs.

8.1.4.1 EVITA HSMs

Three different types of hardware security modules have been defined within the EVITA project: full, medium, and light in order to offer different levels of security functionality and performance.

- The full module is deployed in one or two high-performance communication ECUs in the vehicle, and has hardware for asymmetric cryptographic operations needed by more demanding external communications such as V2X communication. It is proposed to be used only in central communication gateways.
- The medium module is used in two to four central multi-purpose ECUs, such as Gateway ECUs isolating traffic between internal networks (see Section 7.3.2). It supports asymmetric cryptographic operations, but lacks hardware support and is less powerful than the full module.
- The light module is used in less powerful but still security-critical ECUs. It only has a hardware accelerated symmetric cryptographic engine, a hardware random number generator and a UTC clock. Its typical use is in sensors and actuators.

8.1.4.2 Event Data Recorders

Event data recorders (EDR) are devices that record important events and stores them in tamper-proof storage, similar to the black boxes used in aircrafts. It is reasonable to assume that government agencies and vehicle manufacturers will require devices like this to be present in all vehicles, when more advanced applications are introduced. The data recorded in the EDR makes it possible to investigate reasons behind crashes and other safety-critical events.

8.1.5 State-of-the-art Automotive Microcontrollers

An example of a microcontroller for automotive applications is the Freescale MPC5777M Quad-core. Other microcontrollers generally used are from the MPC55XX or MPC56XX series. A detailed list of the Freescale MPC5777M's features taken from its datasheet is shown below:

- **Main features:**
  - Two independent e200z7 cores operating up to 300 MHz
  - Single 300 MHz e200z7 core for delayed lockstep
  - Single e200z4 I/O core operating up to 200 MHz
  - On-chip DSP and floating point unit (on I/O core)
  - Built to support functional safety (ISO 26262 /ASIL-D)
  - 248-channel general timer module (GTM104)
  - Up to 128-channel eDMA
  - Up to 84-channel analog-to-digital converters (ADC)
  - Includes 10 x $\Sigma\Delta$ ADC converters
  - High-speed Nexus Aurora debug and trace support
  - 416-pin PBGA package
  - 512-pin PBGA package
- **Communication Protocols:**
  - Ethernet controller (FEC)
  - 4 x M-CAN and 1 x TT-CAN
  - 6 x LINFlex
  - 8 x dSPI and 2 x IIC
  - 5 x PSI-5 and 15 x SENT
  - Zipwire support
  - 2 x dual-channel FlexRay controller
- **Additional features:**
  - Hardware Security Module (HSM)
  - Tamper Detection Module (TDM)

Figure 8.1 shows a table of current Infineon microcontrollers of the Aurix™series that are suitable for security applications in the automotive sector. Moreover, the Infineon Aurix™microcontrollers fulfil the Evita medium requirements [111].

8.2 OS-level Security

Many current ECUs do not have an OS, due to computational and cost demands. A first step to a common security architecture will be the introduction of a common platform. Automotive Open System Architecture (AUTOSAR) is the solution which is supposed to achieve this unification.

In this Section we will first consider common security mechanisms found in modern operating systems, and then we will compare this to the security features envisioned in the latest AUTOSAR standard.

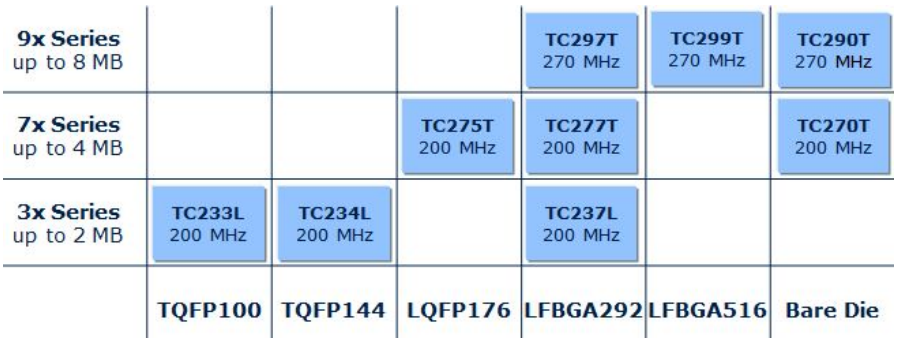

Figure 8.1: Infineon microcontrollers designed for security solutions in the automotive sector (taken from [111]).

8.2.1 Common OS Security Mechanisms

A standard approach to security in operating systems is to use hierarchical protection domains, also known as ring protection. The idea is that outer layers of the ring have a lower privilege level than inner layers. A traditional example of this is the splitting into kernel mode and user mode as in UNIX systems.

Also highly intrinsic to modern operating systems is the separation of process memory. Each process is allocated a specific region in memory, and no process is allowed to access any other region in memory by default.

A more recent approach is to have executable space protection, i.e. to split programs into data and executable portions such that only the executable space is allowed to execute. This is usually accomplished together with hardware support.

Address Space Layout Randomization (ASLR) is another technique that was developed to prevent or at least hinder buffer overflow attacks. This is achieved by randomly arranging the positions of the stack, heap and libraries in the process address space.

Virtualization is already described in Section 8.1.1. A more efficient way of gaining isolation between applications is container-based virtualization. This technique differs from virtualization in the way, that the containers access a single kernel, which results in a more efficient resource usage. Nevertheless this method is not as strong as virtualization with a hypervisor in terms of security for the reason that it is easier to access the entire system. A method to deploy hot updates in a container-based environment on an embedded system is described in [112].

Finally, OS loaders are often signed in order to prevent the execution of untrusted code (see Section 8.1.4). This is known as “secure boot”, and is highly pervasive in game consoles and computer operating systems.

8.2.2 AUTOSAR Security

AUTOSAR's security features, such as the "Specification of Module Secure Onboard Communication" [113] describes the functional specifications for the authentication, verification, and error detection. Moreover this document provides the specification of the API and sequence diagrams of the processes.

AUTOSAR supports the basic process memory protection so that processes can only access their own address space. A Cryptographic Service Manager (CSM) is also supported by AUTOSAR, in order to facilitate the use of different cryptographic algorithms. The diagnostic communication manager (DCM) also knows different security access levels, in order to allow authorized diagnostic procedures.

8.3 Summary

This chapter listed several security mechanisms on hardware and software/OS level. HSMs securely store the private keys, offer cryptographic operations, and have a good random number generator. Additionally, HSMs have to be tamper-proof to ensure that the private keys cannot be stolen or accessed. The three different types of HSMs of the EVITA architecture describe how HSMs can be classified according to their functionality and purpose in the vehicle's system - do they provide hardware accelerated symmetric and asymmetric encryption? Microcontroller that have a built-in HSM are available and some manufacturers even rank their products according to the EVITA project. The use of signature verification of new firmware increases the complexity of the ECUs, but it is necessary for preventing attackers to upgrade an ECU to a modified firmware version.

More complex operating systems have multiple security mechanisms such as a ring layer around the kernel functions, separation of process memory, separation of executable and non-executable storage, booting of only signed firmware, and ASLR. The last mentioned technique hinders the attacker to predict target addresses. However, this method involves also the re-design of diagnostic devices, since they are reading system states from fixed address spaces of an ECU. Virtualization, which is common in the server environment, is approaching also the embedded systems. The OS PikeOS[1], is specially designed for virtualization on microcontroller and supports AUTOSAR. Prospective microcontrollers used in the automotive domain may also include multi-core support to allow running several OSs.

[1] http://www.opensynergy.com/en/products/pikeos/

Chapter 9

Threat Modeling

Threat modeling is a process which helps to identify, enumerate and prioritize potential threats to a software system. It is a systematic analysis of the attacker's profile, attacker's vectors and assets with impact to the organization. This process is often recommended to take place in the early stages of the software development life-cycle and may be repeated several times if necessary. Standards and guidelines, such as ISO 26262 [114] in the automotive safety and SAE J3061 [115] in automotive security domain, have been established to provide guidance during the development of dependable systems. Macher et al. [116] have recently performed a review of threat analysis and risk assessment (TARA) methods in the automotive context. Their main findings identify four most applicable TARA methods (EVITA, HEAVENS, SAHARA and BRA) for early development phase analysis of the system.

The purpose of this chapter is to introduce a categorized list of threat modeling methodologies and techniques. We categorize them below and motivate our choice, respectively. Furthermore, we present characteristics of several threat modeling methodologies in Table 9.1 and TARA methods in Table 9.2. A more complete and systematic list of threat modeling and assessment methodologies will be provided in our future work as part of a systematic literature review.

9.1 Asset-driven threat modeling

Asset-driven threat modeling paradigm involves identifying and analyzing threats to organization's assets. If the threats to core assets are not identified and mitigated, the exploitation of potential vulnerabilities may cause a great loss of resources for the organization. Therefore, determining the threats to core assets goes hand in hand with risk analysis. We continue to list several methodologies as asset-driven and pinpoint the asset-driven threat modeling aspects.

The CORAS methodology [117] for risk analysis consists of a language, a tool and a method. The methodology employs CORAS threat diagrams that describe the threats, vulnerabilities, scenarios and incidents of relevance for the risk in question.

Similarly to the aforementioned CORAS methodology, PASTA [118] is a risk-centric threat modeling methodology. In PASTA threats are analyzed with the help of use cases and Data Flow Diagrams (DFDs). Note that further steps are taken for detailed analysis, namely the use of attack trees and abuse cases, which may suggest commonalities with the attack-centric threat modeling methodologies.

OCTAVE [119] is a risk-based strategic assessment and planning methodology for security with asset-driven characteristics. The first phase of OCTAVE aims towards building asset-based threat profiles, which mainly consists of identification and analysis of core assets and corresponding threats.

SREP [120] is an asset-based and risk-driven method for security requirements elicitation during the development of Information Systems. It is based on the integration of the Common Criteria (CC) into the software development life-cycle. In contrast to the visual CORAS methodology, SREP is a textual method. Centered around critical assets and security objectives, this method retrieves threats in a textual or tabular form.

The AEGIS [121] secure software engineering method elicits security requirements by identifying the assets and analyzing risk and threats in relation to the context of use. This method proposes modeling assets to gather security requirements from stakeholders through abuse cases.

A framework for assessing the security of the connected car infrastructure was introduced by Kleberger et al. in [6]. The framework includes a model of the infrastructure and a security assessment tree, according to which the security issues are identified for two scenarios: remote diagnostics performed by repair shop and remote diagnostics performed by the automotive company applications' centre.

Unlike the methodologies mentioned above, data-centric approaches for threat modeling may revolve around data that is not considered a core asset. Such is the case for the approach developed by Mathews et al. [122], which models user access patterns, upon which statistical learning algorithms are trained for insider attack detection in database systems.

9.2 Attack-centric threat modeling

Attack-centric threat modeling paradigm involves identifying and analyzing threats to a system from the attacker's perspective. It includes profiling attacker's characteristics, level of skillfulness and motivation to exploit vulnerabilities. A mitigation strategy is adopted based on the knowledge gained by threat modeling with misuse cases (MUC), abuse cases, MUC maps, misuse sequence diagrams, attack trees and scenarios, etc. Below we continue to categorize several methodologies as attack-centric.

Secure Tropos [123] is an extension to the agent-oriented Tropos methodology. Instead of treating security requirements as non-functional requirements and modeling them as soft-goals, Secure Tropos includes the concept of constraints with respect to security.

Similarly to Secure Tropos, KAOS is a goal-oriented requirements engineering methodology. It allows requirements to be elicited from goal diagrams. Moreover, KAOS allows obstacle threat analysis using threat trees [124].

Problem frames were introduced by Jackson [125] as a kind of patterns that intuitively identify a class of problems in terms of its context and the characteristics of its domains, interfaces and requirements. In particular, Hatebur and Heisel [126] presented four security frames together with corresponding architectural patterns. Security frames are a complementary approach to eliciting anti requirements and the corresponding abuse frames, presented in [127].

9.3 Software oriented threat modeling

We categorize threat modeling methodologies that are not attack-centric or asset-driven as software oriented. We have identified STRIDE and TRIKE as software oriented threat modeling methodologies. The later offers an open source tool which has seen several improvements, yet in this document we refer to the initially developed and well documented methodology.

STRIDE introduces a set of methods centered around DFDs and a threat enumeration technique called “STRIDE per element”.

	Verification	Modeling technique	Information analysis	Tool support	Validation	Domain	Source
Asset-driven approaches							
CORAS	-	threat, risk diagrams	expert-based	yes	IE, ICS, ACS	non restrictive	[117, 128]
PASTA	-	UC, DFD, attack trees, abuse cases	expert-based	-	IE	non restrictive	[118]
OCTAVE	-	threat profiles	expert-based	-	IE, ICS, ACS	non restrictive	[119, 129]
SREP	-	text-based	knowledge-based	-	ACS, ICS	non restrictive	[120]
AEGIS	-	abuse cases	expert-based	-	ICS	non restrictive	[121]
Framework by Kleberger et al.	-	assessment tree	expert-based	-	ICS	automotive	[6, 130, 131]
Attack-centric approaches							
Secure Tropos	yes	attack scenarios	expert-based	yes	ICS	agent-oriented development	[124, 123]
KAOS	yes	goals, anti goals, anti requirements, obstacle threat analysis	expert-based	yes	ACS, ICS	non restrictive	[124, 132, 133]
Abuse frames	-	anti requirements, abuse frame diagrams	expert-based	-	ACS	non restrictive	[127, 134]
Security frames	-	SPF, CSPF	knowledge-based	-	ACS	non restrictive	[126]
Software oriented approaches							
STRIDE	-	DFD, STRIDE model	expert-based	yes	IE, ICS, ACS	non restrictive	[135]
TRIKE	-	attack trees, graphs	partly knowledge-based	yes	-	non restrictive	[136]

Table 9.1: Main characteristics of threat modeling methodologies (1)
Note: ACS = academic case studies, ICS = industrial case studies, IE = illustrative example

	Verification	Modeling technique	Information analysis	Tool support	Validation	Domain	Source
TARA approaches							
EVITA	-	attack trees	expert-based	-	ICS	automotive	[4]
TVRA	-	threat agents, trees	expert-based	yes	ICS, ACS	telecom networks	[137]
HEAVENS	-	STRIDE model	expert-based	-	ICS	automotive	[138]
FMEA	-	threat agents, STRIDE model	expert-based	yes	IE, ICS, ACS	non restrictive	[139]
SAHARA	-	STRIDE model	expert-based	-	ICS	automotive	[140]

Table 9.2: Main characteristics of TARA methods (2)
Note: ACS = academic case studies, ICS = industrial case studies, IE = illustrative example

Chapter 10

Secure Design

Building secure software solutions requires careful consideration of security related activities throughout the software development life-cycle (SDLC). In this chapter, we focus on activities in the design stage that aim to improve the security of the software solution. We consider 3 publications that thoroughly review secure design methodologies from different perspectives.

Uzunov et al. [141] performed a survey where they analyzed the secure design methodologies applicable to distributed systems. Their work provides detailed descriptions and discussions of selected methodologies, followed by an evaluation. The set of criteria used for evaluating the methodologies comprises of 12 criterion accompanied by corresponding descriptions. The criteria were further categorized into: essential criteria for industry projects, criteria related to qualifying risk associated with adopting a methodology and criteria representing desirable features of a methodology. Based on the detailed reviews, the authors analyzed the conformance of individual methodology to the criteria. However, the authors mention that their analysis may reflect a subjective assessment.

In comparison to the survey mentioned above, Nguyen et al. [142] narrow the scope of secure design methodologies by focusing only on model-driven development of secure systems. In this work, the authors systematically review the literature and identify a set of primary model-driven security studies. The selected studies are evaluated according to the following set of criteria: security concerns, modeling approaches, model-to-model transformations (MMTs), model-to-text transformations (MTTs), application domains and evaluation methods.

Design notations allow the design of a system to be precisely described. Van Berghe at al. [143] have made an extensive systematic literature review on design notations for secure software in the past 10 years. The authors created a structured assessment criteria, where the primary focus was set on assessing the applicability, support for security, purpose, usage and validation of the identified design notations.

As previously mentioned, the publications above review different methodologies. Our aim is to answer the following question for each publication: What does publication *A* reveal about methodology *X*, that other two publications don't? Hence, an overlapping set of secure design methodologies was identified. For the purpose of presenting what the individual publications reveal about each methodology, we focus on the satisfied criterion (e.g. methodology *X* provides tool support). Furthermore, we only take into account the fundamentally different criteria, in order to avoid duplicated table entries. For instance, if publication *A* evaluated methodology *X* based on the criterion of tool support, and publication *B* evaluated methodology *X* on the same criterion, we do not consider it. We only mention the criterion if there is inconsistency in the results of publications.

Tables 10.1 and 10.2 present either noted inconsistencies or additional information about individual methodologies. Some inconsistencies occur due to the use of a different taxonomy. For example, Nguyen et al. consider non-repudiation also a security concern, whereas Van Berghe at al. consider cryptography as one of the security concerns. Several differences were also identified in the presented results regarding validation of SECTET, SecureUML and SecureSOA. This might be due to the fact that Nguyen et al. refer to a larger amount of publications while analyzing these methodologies. Another possible explanation might be that Van Berghe et al. focus on reviewing specific design notations, while Nguyen et al. review Model-driven security approaches. Finally, several criteria were observed to be fundamentally different. The information obtained by observing the values of such criteria provides additional information about the studied methodologies. A good example of such information is the type of Model-to-model transformations supported by UMLSec, which was only explicitly observed by Nguyen et al.

We also provide a more complete list of methodologies identified by the aforementioned publications in Tables 10.3, 10.4, 10.5 and 10.6. In the presented tables we group the methodologies into significant Model-driven security approaches, Pattern-based approaches, Security at runtime, Secure SOA and Aspect-oriented approaches. A set of useful characteristic are extracted for each methodology, such as verification, level of automation, modeling language, tool support, sources for further reading, etc. Note that these tables do not represent the complete union of all studied approaches.

	Survey by Uzunov et al.	SLR by Nguyen et al.	SLR by Van Berghe at al.
UMLSec	partly satisfies criterion that a full spectrum of security related activities are in all SDLC stages	security concerns: C, I, authentication, authorization, non-repudiation; MMT: endogenous; MTT: yes; application domain: web applications, embedded systems, distributed systems	security concerns: C, I, access control, authentication, cryptography; representation support of all security concerns by construction; annotation of design models; verification includes model checking and theorem proving
SECTET	-	security concerns: C, I, authorization, non-repudiation; MMT: exogenous; MTT: yes; application domain: e-government, e-health, e-education, web services; validation: ACS	security concerns: auditability, access control, cryptography; representation support of security concerns by construction; annotation of design models or providing separate models; validation: illustrations
SecureUML	not included in the study	partly supports AOM and SOC: MMT: endogenous; MTT: yes; verification: SecureMora model-checker; validation: illustrations, ACS, ICS; main limitation is sole focus on access control	representation support of security concerns by construction; annotation of system design models; verification: OCL; validation: illustrations
ModelSec	-	security concerns: C, I, A, authentication, authorization; MMT: exogenous, MTT: partly; application domain: non-restricted; validation: ACS	not included in the study

Table 10.1: Comparison of overlapping methodologies (1)

Note: MMT = Model-to-model transformations, MTT = Model-to-text transformations, endogenous = transformations within one metamodel, exogenous = transformations between different metamodels, C = confidentiality, I = integrity, A = availability, ACS = academic case studies, ICS = industrial case studies, AOM = Aspect-Oriented modeling, SOC = Service-Oriented computing, OCL = Object Constraint Language

	Survey by Uzunov et al.	SLR by Nguyen et al.	SLR by Van Berghe at al.
Aspect-Oriented (Georg)	provides guidelines to balance security with other qualitative attributes	security concerns: C, I, A, authorization, authentication; supports AOM and SOC; MMT: endogenous; MTT: not provided;	security concerns: authentication; support of security concerns by documentation; verification: OCL and Alloy
Aspect-Oriented (Mouheb)	-	security concerns: C, authorization; MMT: endogenous; MTT: provided; verification: none; non-restricted domain of application; validation: ACS	not included in the study
SecureSOA	not included in the study	security concerns: C, I, A, authorization; validation: ACS	security concerns: C, I, A, cryptography; support of security concerns by documentation; annotation of system design models; validation: illustrations

Table 10.2: Comparison of overlapping methodologies (2)

Note: MMT = Model-to-model transformations, MTT = Model-to-text transformations, endogenous = transformations within one metamodel, exogenous = transformations between different metamodels, C = confidentiality, I = integrity, A = availability, ACS = academic case studies, ICS = industrial case studies, AOM = Aspect-Oriented modeling, SOC = Service-Oriented computing, OCL = Object Constraint Language

	Verification	Automation	Modeling language	Tool support	Validation	Domain	Source
Significant MDS approaches							
SECTET	-	MMT	UML	yes	ACS	e-government, e-health, e-education, web services, SOA	[144, 145, 146, 147, 148, 149, 150]
Secure-DWs	-	MMT	UML	prototype tool	IE, ACS	web applications, databases	[151, 152, 153, 154, 155, 156, 155, 157, 158, 159, 160, 161]
Secure-MDD	KIV theorem prover, test cases from UML specifications	MMT and MTT	UML	-	IE, ACS, ICS	smart card and service platforms	[162, 163, 164, 165]
Secure-UML	Secure-Mova model-checker	MMT, partly MTT	UML	yes	IE, ACS, ICS	web applications, embedded systems, distributed systems	[166, 167, 168, 169, 170, 171, 172, 173]
UMLsec	AICALL theorem prover	MMT	UML	yes	IE, ACS, ICS	web applications, embedded systems, distributed systems	[174, 175, 176, 177, 178]

Table 10.3: Main characteristics of secure design methodologies (1)

Note: MMT = Model-to-model transformations, MTT = Model-to-text transformations, endogenous = transformations within one metamodel, exogenous = transformations between different metamodels, C = confidentiality, I = integrity, A = availability, IE = illustrative example, CE = controlled experiment ACS = academic case studies, ICS = industrial case studies, AOM = Aspect-Oriented modeling, SOC = Service-Oriented computing, OCL = Object Constraint Language

	Verification	Automation	Modeling language	Tool support	Validation	Domain	Source
			Pattern-based				
Abramov et al.	yes	MMT and MTT	UML	yes	ACS, CE	database	[179, 180]
Bouaziz et al.	-	MMT and MTT	UML	-	ACS	component-based architecture	[181]
Kim et al.	yes	MMT	UML	-	ACS	non restrictive	[182, 183]
Schnjakin et al.	-	MMT	BPMN	yes	IE	service-oriented architectures	[184]
Moral et al.	-	MMT	DSL	-	IE	secure cloud computing	[185]
PSecGCM/ SecMob-Grid	yes	-	UML	yes	ICS	mobile grid systems	[186, 187]
SEPP	yes	-	partly UML	yes	ACS	non restricted	[188, 189, 190, 191, 192, 193]
SERENITY	yes	-	UML	yes	ICS	non restrictive	[194, 195]
PWSSec	yes	-	UML	partly	ICS	web services	[196, 197]

Table 10.4: Main characteristics of secure design methodologies (2)

Note: MMT = Model-to-model transformations, MTT = Model-to-text transformations, endogenous = transformations within one metamodel, exogenous = transformations between different metamodels, C = confidentiality, I = integrity, A = availability, IE = illustrative example, CE = controlled experiment ACS = academic case studies, ICS = industrial case studies, AOM = Aspect-Oriented modeling, SOC = Service-Oriented computing, OCL = Object Constraint Language

	Verification	Automation	Modeling language	Tool support	Validation	Domain	Source
			Sec@runtime				
SecDSVL, MDSE@R	testing	MMT, partly MTT	UML	yes	ACS, CE	cloud-based applications	[198, 199]
Elkrakaiby et al.	-	MMT	UML	prototype tool	ACS	non restrictive	[200]
Morin et al.	-	MMT, partly MTT	DSL	-	ACS	component-based architecture	[201]
			SecureSOA				
Gilmore et al.	yes	MMT and MTT	UML	-	ACS	distributed systems	[202]
Menzel et al.	-	MMT, partly MTT	DSL	yes	ACS	SOA	[203, 204]
Wolter et al.	-	MMT, partly MTT	DSL	yes	IE	SOA	[205]
Nakamura et al.	-	MMT, partly MTT	UML	-	IE	SOA	[206]
			Aspect-oriented security				
Georg et al.	yes	MMT	UML	prototype tool	IE	non restrictive	[207]
Mouheb et al.	-	MMT and MTT	UML	yes	ACS	non restrictive	[208]
Ray et al.	-	MMT	UML	-	IE	non restrictive	[209]
Sanchez et al., ModelSec	-	MMT, partly MTT	UML	yes	ACS	non restrictive	[210]

Table 10.5: Main characteristics of secure design methodologies (3)

Note: MMT = Model-to-model transformations, MTT = Model-to-text transformations, endogenous = transformations within one metamodel, exogenous = transformations between different metamodels, C = confidentiality, I = integrity, A = availability, IE = illustrative example, CE = controlled experiment ACS = academic case studies, ICS = industrial case studies, AOM = Aspect-Oriented modeling, SOC = Service-Oriented computing, OCL = Object Constraint Language

	Verification	Automation	Modeling Language	Tool support	Validation	Domain	Source
Access control oriented							
Ahn et al.	-	partly MTT	UML	prototype tool	CE	non restrictive	[211]
Kim et al.	partly	MMT and MTT	UML	prototype tool	ACS	non restrictive	[212]
Sohr et al.	-	not provided	UML	prototype tool	ACS	distributed systems	[213]
Koch et al.	graph analysis	-	UML	-	IE	non restrictive	[214, 215]
Mariscal et al.	-	-	UML	prototype tool	IE	non restrictive	[216]
Yu et al.	yes	-	UML	-	IE	non restrictive	[217]
Burt et al.	-	MMT	DSL	-	IE	non restrictive	[218]
Fink et al.	-	MMT	DSL	-	ACS	non restrictive	[219]
Mouelhi et al.	yes	MMT, partly MTT	DSL	-	ACS	non restrictive	[220]
Kaddani et al.	-	MMT, partly MTT	UML	-	IE	Electrical grid	[221]
Bertolino et al.	Model-based testing	MMT, partly MTT	DSL	yes	ACS	non restrictive	[222]
Eby et al.	-	MMT	DSL	yes	ACS	embedded systems	[223]
ADM-RBAC	-	-	non UML diagrams	prototype tool	IE, CE	web services	[224]

Table 10.6: Main characteristics of secure design methodologies (4)

Note: MMT = Model-to-model transformations, MTT = Model-to-text transformations, endogenous = transformations within one metamodel, exogenous = transformations between different metamodels, C = confidentiality, I = integrity, A = availability, IE = illustrative example, CE = controlled experiment ACS = academic case studies, ICS = industrial case studies, AOM = Aspect-Oriented modeling, SOC = Service-Oriented computing, OCL = Object Constraint Language

Chapter 11

Secure Software Development Lifecycle (SDL) and Processes

In this chapter, we briefly present state-of-the-art in the area of Security Development Lifecycle (SDL) and processes with focus on software development. Although most SDLs and processes are not specific to the automotive Electrical and/or Electronic (E/E) systems, those can potentially be adapted to meet the needs and requirements of the automotive industry in future. SDL is a security assurance process that is focused on software development. Combining a holistic and practical approach, the SDL introduces security and privacy throughout all phases of the development process to reduce the number and severity of software vulnerabilities [225]. In the past, application development, and more specifically the coding associated with software development, was a somewhat separate function; functional requirements were presented to developers and a finished application was then turned out that satisfied those business requirements. Today, it is much more complex and sophisticated for a myriad of reasons. This increased complexity makes the need for a practical approach to developing software in a secure manner even more necessary [225].

11.1 ISO 27034

ISO recently published a new standard as part of its information security-focused 27000 series that focuses on application security. ISO/IEC 27034 provides guidance to assist organizations in integrating security into the processes used for managing their applications. It introduces definitions, concepts, principles and processes involved in application security [226]. The purpose of ISO/IEC 27034 [226] is to assist organizations in integrating security seamlessly throughout the life cycle of their applications by:

1. providing concepts, principles, frameworks, components and processes;

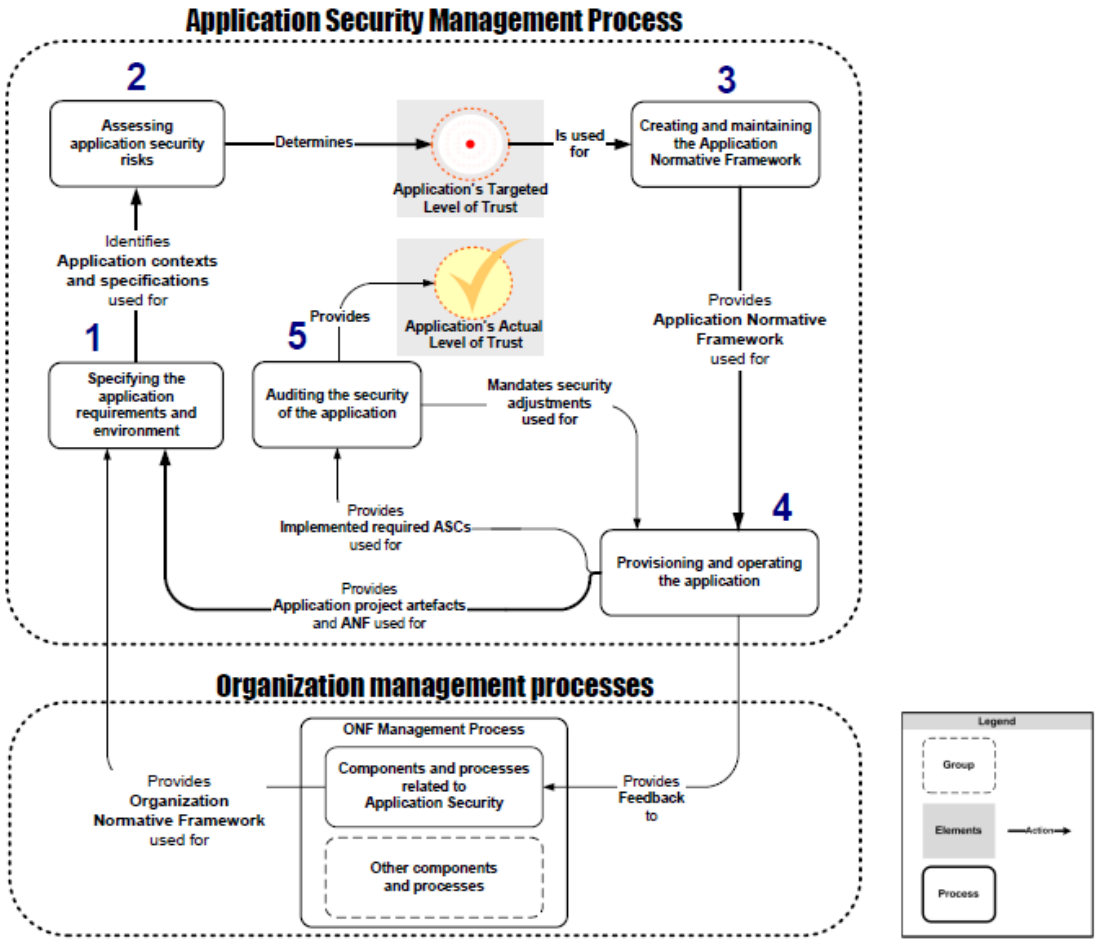

Figure 11.1: ISO Application Security Management Process

2. providing process-oriented mechanisms for establishing security requirements, assessing security risks, assigning a Targeted Level of Trust and selecting corresponding security controls and verification measures;

3. providing guidelines for establishing acceptance criteria to organizations outsourcing the development or operation of applications, and for organizations purchasing from third-party applications;

4. providing process-oriented mechanisms for determining, generating and collecting the evidence needed to demonstrate that their applications can be used securely under a defined environment;

5. supporting the general concepts specified in ISO/IEC 27001 and assisting with the satisfactory implementation of information security based on a risk management approach; and

6. providing a framework that helps to implement the security controls specified in ISO/IEC 27002 and other standards.

Figure 11.1 is extracted from ISO/IEC 27034-1 and shows the processes associated with application security management.

11.2 CISCO Secure Development Lifecycle (CSDL)

Figure 11.2: CISCO Secure Development Lifecycle

The Cisco Secure Development Lifecycle (CSDL) is a repeatable and measurable process to increase the resiliency and trustworthiness of CISCO products [227]. CSDL:

- Uses industry-leading technology and practices
- Applies across multiple operating systems
- Adapts to Agile and Waterfall development methods
- Is part of Cisco Product Development Methodology (PDM) and ISO9000 compliance requirements
- Benefits customers who deploy high-quality products they can trust

While each Cisco SDL element helps maximize protection against a specific set of critical exploits, no single element can ensure a product is “safe.” The combination of tools and processes introduced in all phases of the development lifecycle helps to ensure in-depth defense and provides a holistic approach to product resiliency. Figure 11.2 shows the CSDL developed and deployed by CISCO.

11.3 McAfee Security Development Lifecycle

Integrating security early into the application development lifecycle produces more secure, robust applications at a lower cost [228]. Figure 11.3 shows the secure software development lifecycle as proposed by Foundstone.

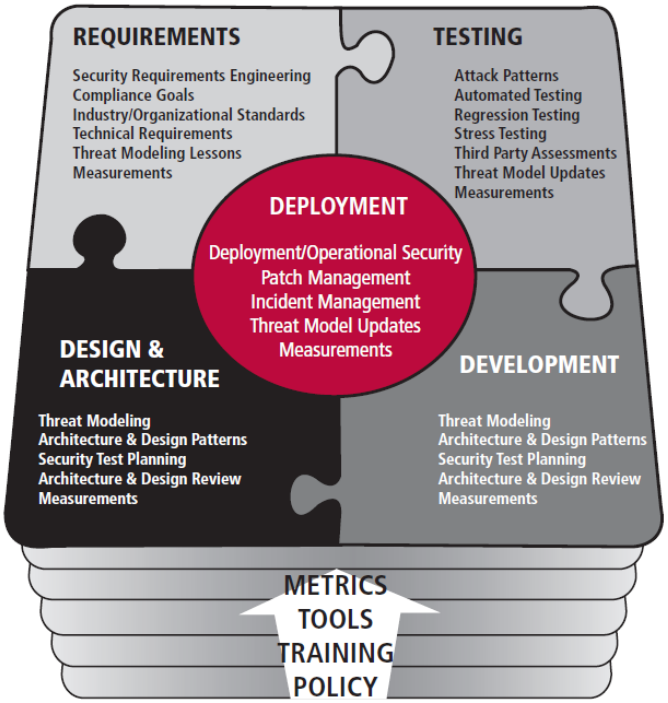

Figure 11.3: McAfee Secure Development Lifecycle

11.4 Microsoft

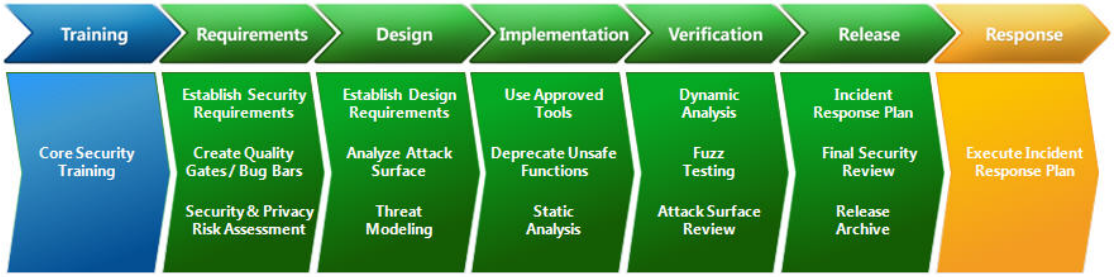

Figure 11.4: Microsoft Secure Development Lifecycle

Developed by Microsoft, the SDL is a software development security assurance process consisting of security practices grouped by seven phases: training, requirements, design,

implementation, verification, release and response [229]. Microsoft provides a large number of SDL templates, tools and other resources at no cost to assist other companies wishing to improve the security of the software they develop. Figure 11.4 shows the SDL developed and used by Microsoft.

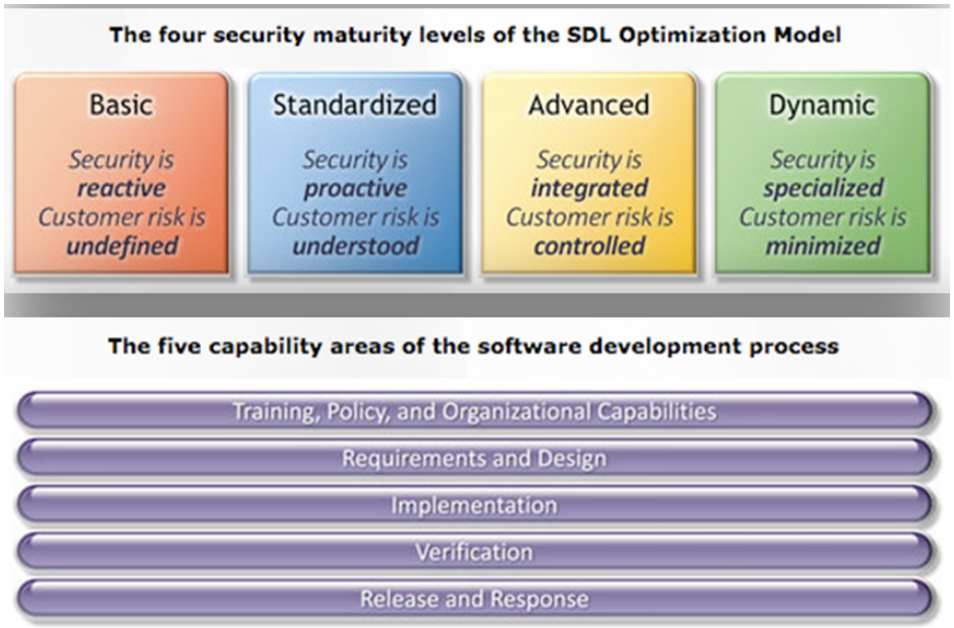

Figure 11.5: Microsoft Optimized Secure Development Lifecycle Model

The quantitative and qualitative benefits of using the Microsoft SDL range from reduced development costs and time, to more secure applications that leverage the critical training and knowledge across the entire development organization [225]. No longer is it necessary to teach each developer the entire realm of knowledge to be able to manually develop secure applications. Also, it reduces the need to depend on each programmer to purposely program using secure coding practices. Instead, the infrastructure and the overall development system enforces secure coding through the development methodology and confirms it through both manual and automated testing [225]. Many industries have adopted the SDL proposed by Microsoft. For example, Microsoft SDL has been integrated into the Software Design Life Cycle (SDLC) of financial services companies. Based on Microsoft SDL, BITS, the technology policy division of the Financial Services Roundtable, published a Software Assurance Framework to provide an overview of the components of a mature, strategic software development program for financial institutions [229]. Also, it has been demonstrated how the Microsoft SDL can help meet some of the requirements of the Payment Card Industry Data Security Standard (PCI DSS) and the Payment Application Data Security Standard (PA-DSS) [229]. Furthermore, it has been shown how the Microsoft SDL can help organizations comply with some requirements of the administrative simplification provision of the Health

Insurance Portability and Accountability Act and its implementing regulations (HIPAA), including the Security Standards for Protecting Electronic Protected Health Information (HIPAA Security Rule) and the Standards for Privacy of Individually Identifiable Health Information (Privacy Rule), as well as the American Recovery and Reinvestment Act of 2009 (ARRA), particularly Title XIII of ARRA, called the Health Information Technology (HIT) for Economic and Clinical Health (HITECH) Act [229]. It has been shown how SDL practices and HIPAA requirements intersect in very practical ways by using two common scenarios in the healthcare software ecosystem: Developing new software and Integrating new software modules or interfaces for a medical environment [229]. Subsequently, Microsoft proposes an optimized SDL model to make SDL adoption easier. Figure 11.5 shows the optimized SDL model.

11.5 Software Assurance Maturity Model (SAMM)

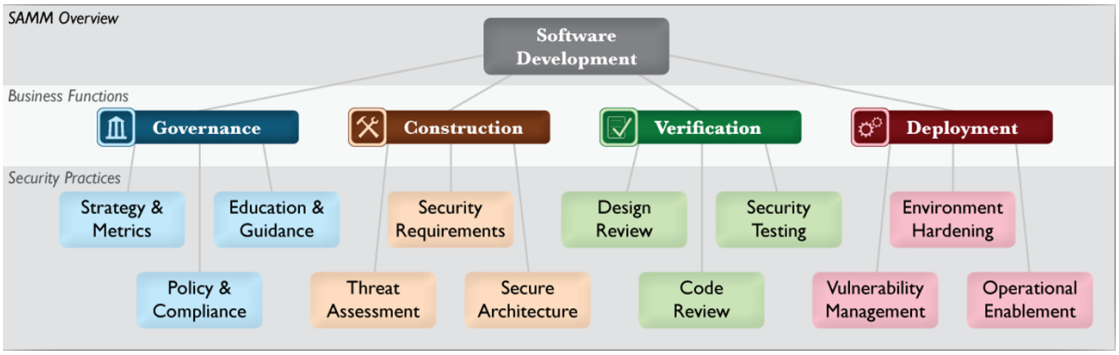

Figure 11.6: Overview of the Software Assurance Maturity Model (SAMM)

The Software Assurance Maturity Model (SAMM) is an open framework to help organizations formulate and implement a strategy for software security that is tailored to the specific risks facing the organization [230]. The foundation of the model is built upon the core business functions of software development with security practices tied to each (see diagram below) [230]. The building blocks of the model are the three maturity levels defined for each of the twelve security practices [230]. These define a wide variety of activities in which an organization could engage to reduce security risks and increase software assurance. Additional details are included to measure successful activity performance, understand the associated assurance benefits, estimate personnel and other costs [230]. As an open project, SAMM content shall always remain vendor-neutral and freely available for all to use. Figure 11.6 depicts an overview of this model.

11.6 Building Security In Maturity Model (BSIMM)

The Building Security In Maturity Model (BSIMM) is the result of a multi-year study of real-world software security initiatives and presents the model as built directly out of data observed in sixty-seven software security initiatives [231]. The BSIMM is a measuring stick for software security. The best way to use the BSIMM is to compare and contrast

your own initiative with the data about what other organizations are doing contained in the model [231]. You can then identify goals and objectives of your own and look to the BSIMM to determine which additional activities make sense for you [231]. The BSIMM data show that high maturity initiatives are well rounded—carrying out numerous activities in all twelve of the practices described by the model [231]. The model also describes how mature software security initiatives evolve, change, and improve over time [231]. Figure 11.7 presents overview of BSIMM.

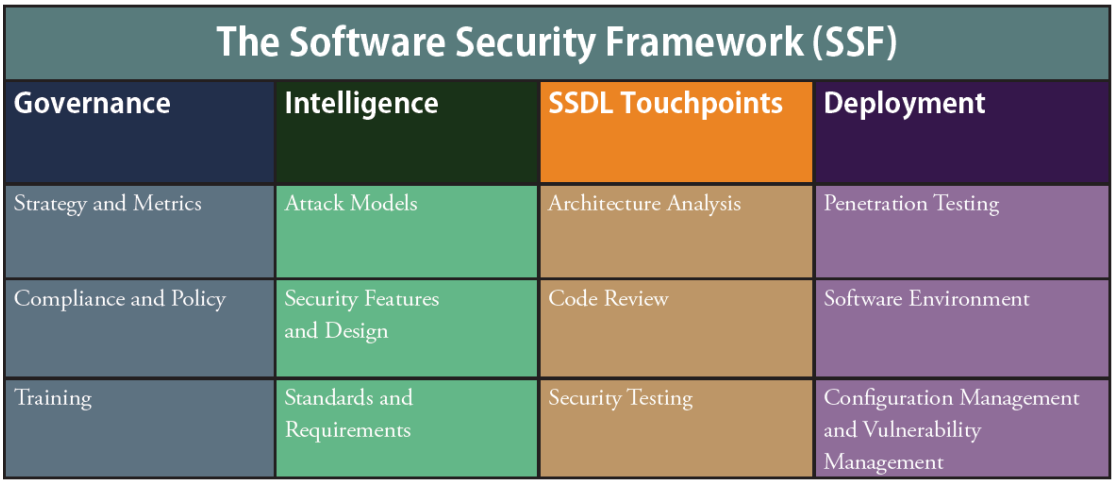

Figure 11.7: Overview of the Building Security In Maturity Model (BSIMM)

11.7 The Comprehensive, Lightweight Application Security Process (CLASP)

The Comprehensive, Lightweight Application Security Process (CLASP) provides a well-organized and structured approach for moving security concerns into the early stages of the software development lifecycle, whenever possible [232]. CLASP is actually a set of process pieces that can be integrated into any software development process and designed to be both easy to adopt and effective [232]. It takes a prescriptive approach, documenting activities that organizations should be doing. And, it provides an extensive wealth of security resources that make implementing those activities reasonable [232]. Figure 11.8 shows the CLASP security model.

11.8 Tools for software security testing and evaluation

Virtually every SDL that focuses on software development suggests applying code review and static analysis as means of detecting security defects early in the development lifecycle. Furthermore, SDLs recommend performing threat modeling and risk assessment during requirement and design phases as well as dynamic testing during verification phase. In

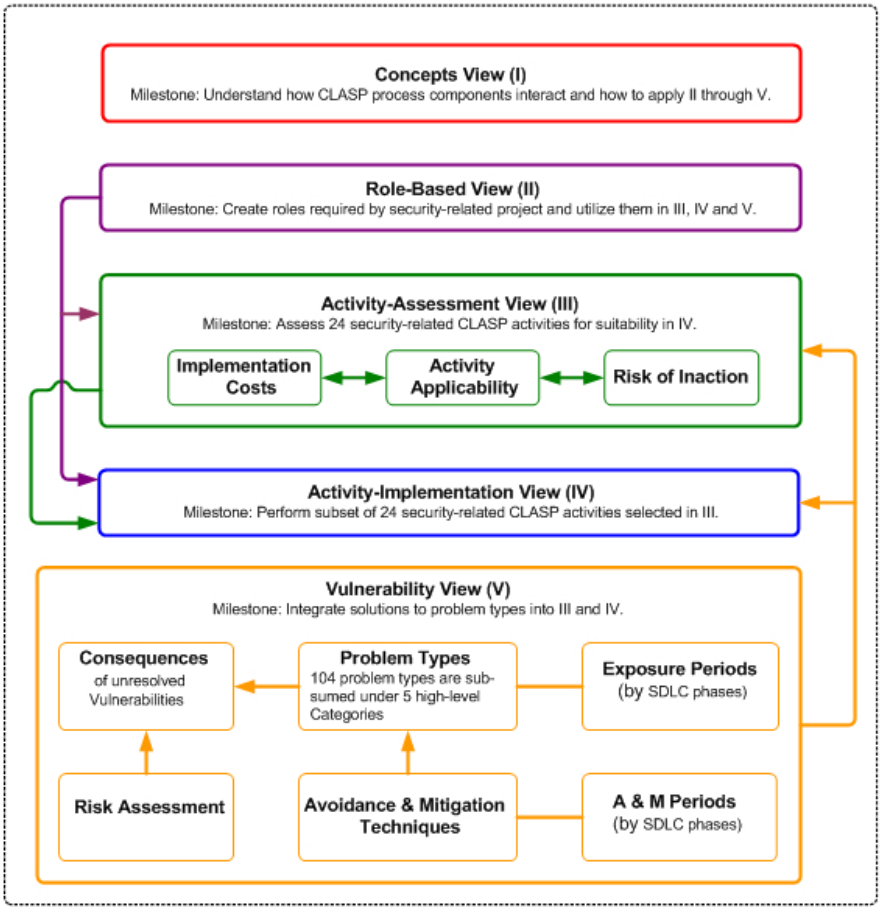

Figure 11.8: CLASP Security Model

this sub-section, we briefly present several tools that can be used during different phases of the SDL.

11.8.1 Static analysis

Static analysis is the process of examining the source code of a program and in that way tests a program for various weaknesses without having to actually execute it (compared to dynamic analysis such as testing). The thesis work by Patrik Hellström [233] has investigated what different tools for static code analysis, with an emphasis on security, there exist and which of these that possibly could be used in a project at Ericsson AB in Linköping in which a HIGA (Home IMS Gateway) is constructed. Today there exist a number of different vendors of static analysis tools and both commercial as well as open source tools are available. In this sub-section, we extract a list of commercial static analysis tools as presented by Patrik Hellström [233] and extend descriptions to represent the current state-of-the-art:

11.8.1.1 Fortify

Fortify Software Inc. was founded in 2003 and has since then provided their customers with tools for finding security vulnerabilities in software applications. The Fortify Source Code Analyzer (SCA) is the tool to perform static analysis. It can be run on most operating systems and provide functionality for scanning C/C++, Java, .Net, PL-SQL, T-SQL and ColdFusion code [233]. Fortify company is acquired by HP and the tool is now known as HP Fortify Static Code Analyzer. Fortify has taken two top honors in 2012 in the "Best of Application Security" category with Fortify Real-Time Analyzer (now Fortify Runtime) taking gold and Fortify Static Code Analyzer taking bronze.

11.8.1.2 Ounce

Ounce Labs was founded in 2002 and their tool Ounce 6.0 (released in July 2008) is a tool that focuses purely on software security. The tool supports both developers with support for integration in different IDEs as well as audit and quality assurance (QA) teams. The tool supports C/C++/C#, Java, VB.NET, ASP.NET and more [233]. The company is acquired by IBM. With this acquisition, Ounce Labs becomes a part of the IBM Rational software business and a critical component of the Rational AppScan portfolio of application security and compliance software.

11.8.1.3 Coverity

Coverity was founded in 2003 in the Computer Systems Laboratory at Stanford University in Palo Alto, California. Coverity Static Analysis automatically tests code as it is being written to help users find high-risk security vulnerabilities prioritized alongside quality defects, in the developer's workflow. Coverity Static Analysis's engine identifies the most critical bugs in C/C++, Java, and C# code bases, scaling to thousands of developers working across geographic boundaries, thousands of defects, and millions of lines of code in a single analysis. Parallel analysis and incremental analysis for C/C++ and Java allow code to be analyzed in minutes in many cases enabling developers to scan their code more frequently and efficiently. It also supports many IDEs in order to be used directly by developers when coding or it can be used as part of the central build system.

11.8.1.4 Klocwork Insight

Klockwork Insight is powered by a comprehensive static analysis engine. Klocwork Insight combines on-the-fly analysis, drag & drop build reporting, and cross-project impact analysis to deliver serious productivity gains to the entire development process. Supported languages include C/C++, Java, and C#. The tool supports a wide variety of IDEs, platforms and builds environments.

11.8.1.5 CodeSonar

GrammaTech's CodeSonar is a relatively new static analysis tool and is used to find vulnerabilities in C/C++ source code. The functionality in CodeSonar makes use of the company's other tool called CodeSurfer [233]. CodeSurfer is a code browser which can be used when performing a manual code review and it is based on research conducted at the University of Wisconsin [233]. It identifies programming bugs that can result in system crashes, memory corruption, and other serious problems. It performs whole-program source code analysis on code bases over 10 million lines of code. And, it includes workflow automation features, like an API for custom integrations and support for extensions that add custom checks, allowing your team to tap its power quickly and completely.

11.8.2 Threat modeling and risk analysis

Microsoft provides SDL Threat Modeling Tool. The tool is publicly available and is developed to be used during the design phase to identify and mitigate potential security issues early, when they are relatively easy and cost-effective to resolve [234]. The SDL Threat Modeling Tool plugs into any issue-tracking system, making the threat modeling process a part of the standard development process [234]. Innovative features include:

- Integration: Issue-tracking systems
- Automation: Guidance and feedback in drawing a model
- STRIDE per element framework: Guided analysis of threats and mitigations
- Reporting capabilities: Security activities and testing in the verification phase

Microsoft also provides a publicly available tool Attack Surface Analyzer.

11.8.3 Dynamic testing (penetration testing and fuzz testing)

11.8.3.1 CERT Basic Fuzzing Framework (BFF)

The CERT Basic Fuzzing Framework (BFF) is a software testing tool that finds defects in applications that run on the Linux and Mac OS X platforms [235]. The BFF automatically collects test cases that cause software to crash in unique ways, as well as debugging information associated with the crashes [235]. The goal of BFF is to minimize the effort required for software vendors and security researchers to efficiently discover and analyze security vulnerabilities found via fuzzing [235].

11.8.3.2 SDL MiniFuzz File and Regex Fuzzer

SDL MiniFuzz File Fuzzer is a basic file fuzzing tool designed to ease adoption of fuzz testing by non-security developers who are unfamiliar with file fuzzing tools or have never used them in their current software development processes [236]. SDL Regex Fuzzer is a

tool to help test regular expressions for potential denial of service vulnerabilities [237]. Regular expression patterns containing certain clauses that execute in exponential time (for example, grouping clauses containing repetition that are themselves repeated) can be exploited by attackers to cause a denial-of-service (DoS) condition. SDL Regex Fuzzer is a tool to help test regular expressions for these potential vulnerabilities during the Verification phase of the Microsoft SDL process [237].

11.8.3.3 Secure software development processes compared

In this subsection, we consider the comparative study by De Win et al. [238], where the authors identified three high profile secure development processes, namely Microsoft's SDL, OWASP's CLASP and McGraw's Touchpoints. In their work they pinpointed the commonalities, discussed the specificity of each approach and suggested improvements. First, the authors accumulated a hierarchical list of activities and merged the ones with similar concepts. Next, they organized the activities into phases of a typical software development process (Education and Awareness, Project Inception, Analysis and Requirements, Architectural Design, Detailed Design, Implementation, Testing, Release and Deployment and Support). Finally, they constructed a structured activity-matrix and used it as the base for their analysis. The authors also included an experimental case study to verify their findings, the description of which goes beyond the scope of this document.

The remainder of this subsection contains a description of the phase-by-phase comparison presented by De Win et al. and subsections describing each approach. We continue to describe findings that the authors presented in the phase-by-phase comparison.

1. Education and awareness
   SDL's training program is impressive and it also puts more effort on the management perspective of education in comparison with other processes. CLASP takes a somewhat different approach to education and awareness as it broadens the audience of training and involves all roles of the project. Touchpoints is more vague about it's goal for education.

2. Project inception
   Attention must be paid to install program-wide, security relevant logistic support such as team communication and bug tracking. Both SDL and CLASP thoroughly address the metrics-challenge and suggest metric related to particular activity. Touchpoint, on the other hand, touches upon the use of metrics, but is more vague in how to realize it in practice. None of the processes provide a systematic integration of metrics within all activities.

3. Analysis and requirements
   Touchpoint covers the broadest spectrum of activities, SDL is restricted to architectural level threat modeling and CLASP is positioned in between. The authors

also point out, that for all processes resource driven threat modeling is based on technical aspects, whereas asset driven threat modeling might be beneficial. Furthermore, the sources used for elicitation of threats are different for all processes. SDL bases threat elicitation on threats to the system. For CLASP additional requirements might originate from the global company policy. In Touchpoints additional requirements may also originate from contractual obligations, laws and regulations.

4. Architectural and detailed design
   SDL is noted to be the most precise and thorough with using STRIDE. However, one drawback of STRIDE is that the number of discovered threats can easily explode. Touchpoints is a bit less complete and CLASP does not provide any guidance about how to carry out threat analysis. Interestingly, both Touchpoints and SDL focus on the security expert and put significant emphasis on risk-driven prioritization of threats, whereas CLASP has a more holistic view and considers several stakeholders.

5. Implementation and testing
   All processes stress the importance of security testing, yet they differ in the approach. SDL mostly adopts the black-hat approach to testing, while CLASP adopts the white-hat approach. Touchpoints was noted to be in between. However, SDL goes a step further with testing as effort is put into tracking the project status for project managers.

6. Release, deployment and support
   Generally, all processes put more focus on the support and less on deployment and implementation. Touchpoints is an exception, as it contains an activity for configuration of access control, logging and monitoring. Overall, CLASP does a good job in detailing the type of documentation to be produced. SDL has a better coverage of the management of security reports and patches.

For further details on the complete comparison, related work, experimental case study and suggestions for improvement, we encourage the reader to refer to De Win et al. [238]. We continue to describe the aforementioned approaches in the following subsections.

11.8.3.4 Microsoft's Security Development Lifecycle (SDL)

As a result of years of research, Microsoft introduced a list of activities organized into 12 stages, aimed to address the security issues often encountered in their process of software development. Below, we list the stages of SDL followed by a brief description extracted from the published documentation of the methodology [239]:

1. *Education and awareness*: Security training is not only encouraged, but is essential and is clearly emphasized as a very important part of software development at Microsoft. Education and awareness stage is a reoccurring activity, as it takes place every year.

2. *Project inception*: This activity comprises of determining whether the application is covered by SDL, assigning a security advisor, building the security leadership team, determining the Bug Bar, etc.
3. *Define and follow Design best Practices*: In order to make secure design more accessible to the average non-security experts, common secure-design principles are followed and attack surface reduction is performed.
4. *Product Risk Assessment*: Before a great amount of effort is put into implementing the software, this activity drives managers to perform the security risk assessment. Depending on the type of data the software is associated with, a so called Privacy Impact Rating is determined.
5. *Risk Analysis*: This activity mainly comprises of the threat modeling process, which is performed by defining scenarios, a list of external dependencies, security assumptions, external security notes, DFDs, threats, determining the risk and finally planning the mitigations.
6. *Creating Security Documents, Tools and Best Practices for Customers*: An effort is made to inform and educate the customers about the security measures adopted and the implications of configuration modification.
7. *Secure Coding Policies*: Several best coding practices are recommended.
8. *Secure Testing Policies*: Activities such as fuzz testing, penetration testing, runtime verification, re-viewing threat models and reevaluating the attack surface are required to be performed.
9. *The Security Push*: The security push targets mainly the some form of legacy code, where security was possibly not the primary concern. It comprises of activities such as training, code reviews, updating threat models, security testing and documentation scrub.
10. *The Final Security Review*: The final security review comprises of activities such as product team coordination, validation and threat models review.
11. *Security Response Planning*: The purpose of this stage is to build a security response center and plan how to handle security issues that may occur in the future.
12. *Product Release*
13. *Security Response Execution*: The core activity is to follow the plan established in early stages.

According to De Win et al., the SDL process is designed to incorporate security as a supporting quality to the functionality-driven software. Moreover, the process is well organized, provides good guidance in terms of activity execution and adopts a management perspective, which complies with the discovery that security needs to be managed, as well as implemented.

11.8.3.5 The Comprehensive, Lightweight Application Security Process (CLASP)

The Comprehensive, Lightweight Application Security Process (CLASP) is a well structured secure software development process which is composed of views, resources and vulnerability use cases. As described in the documentation found online [232], views are further broken down into activities, which contain individual components. In the center of attention are the 24 security-related activities, the subset of which can be fitted into a software development process.

According to De Win et al., the activities of CLASP are defined from a security-theoretical perspective, hence the coverage set of activities is fairly broad. What is more, these activities are independent that have to be integrated in the development process. The choice of the subset and the order of execution is not defined, which results in a higher flexibility of the software development process. On the other hand the authors believe the integration of the later is not always straight-forward.

11.8.3.6 Touchpoints

Touchpoints introduces a set of best practice activities proposed by McGraw in the book Software Security: Building Security In [240]. These lightweight activities should, according to the author, start in the early stages of software development and be revisited multiple times throughout the development process. Below we enumerate these 7 activities, also called *touchpoints*:

1. Code review (Tools)
2. Architectural Risk Analysis
3. Penetration testing
4. Risk-based Security Testing
5. Abuse Cases
6. Security Requirements
7. Security Operations

Figure 11.9 shows the seven touchpoints ordered by the author according to effectiveness and importance. Risk analysis is a reoccurring activity in Touchpoints, which expresses the importance of risk management in secure software development process. Touchpoints provides a set of Black- (e.g. penetration testing) and White-hat (e.g. static code review) activities, both of which are required for good results. Similarly to CLASP, Touchpoints activities can be tailored to the software development process of individual companies.

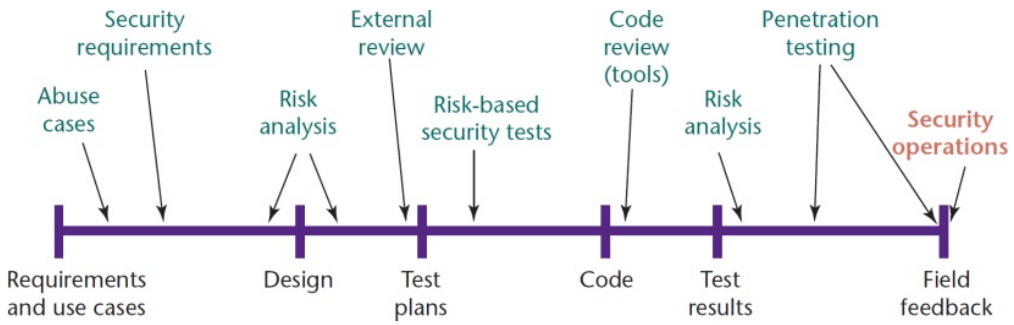

Figure 11.9: A set of best practice activities, also called touchpoints. The figure is taken from [241].

Chapter 12

Evaluation and Certification of Security Functionality

12.1 A Framework for assessing Security

Although there is a lot of research going on in vehicular systems, we have found very little research referring to models of the connected car and how to assess the security of emerging vehicle services, e.g. remote diagnostics, remote software download, and other internet services brought into future vehicles.

The Car-2-Car Communication Consortium (C2C-CC) have created a reference architecture which is divided into three domains; the *in–vehicle*, the *ad hoc* and the *infrastructure* domains [242]. The in-vehicle domain is represented by the vehicle, its applications and mobile devices directly associated to the vehicle. The ad hoc domain is represented by the vehicles and the RSUs, where the RSUs further can be connected to the infrastructure domain. In their architecture, the access network, the internet, and possible nodes connected to the internet are shown as part of the infrastructure domain.

A more detailed framework for security assessment has been developed in [6] which consists of *a model* for the infrastructure of the connected car and *a security assessment tree*. Such a framework can help to understand and evaluate how secure protocols and applications should be evaluated and designed in different vehicle settings [243, 244, 245]. The proposed model together with the security assessment tree makes it easier to identify the weaknesses of the system and the existence of threats both when designing new services and when assessing security as a whole.

This model is shown in Figure 12.1. *The infrastructure* is divided into two domains, the managed infrastructure and the vehicle communication domain. The managed infrastructure is further divided into five regions: automotive company applications' center, third party applications' center, trusted network, untrusted network, and the internet

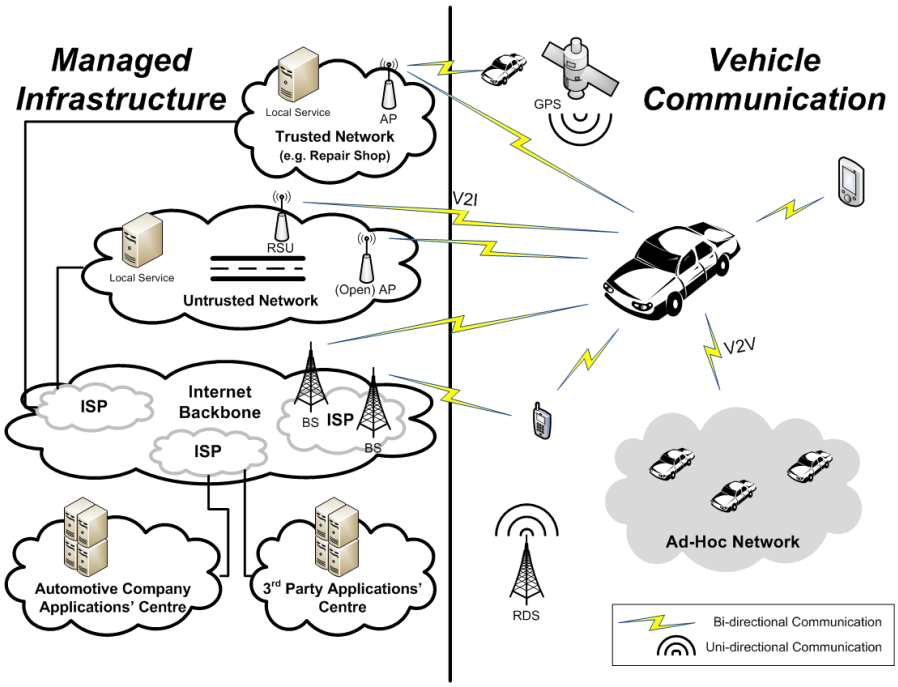

Figure 12.1: Communication scenarios and trust relationships

backbone. *The vehicle communication* describes the possible means of communication with the vehicle. These concepts are further explained in the following paragraphs.

Kiening et al. propose in [246] the use of trust assurance levels. These levels are used to verify the protection level of the sender. The levels are from 0 to 4 and indicate the trust in the information. Thus, a node is able to use the received information combined with the knowledge of the trust assurance level. For instance, safety critical information has to be provided with a certain trust assurance level in order to be further processed. A level of 0 implies that no security mechanisms have been used. 1 entails that software mechanisms for security are implemented. Level 2 involves hardware and software security mechanisms. The security of directly connected components is guaranteed in level 3 and level 4 implies the protection of all components involved in V2X applications [246].

12.1.1 Managed Infrastructure

The five regions of the managed infrastructure show different levels of trust which may require different protection mechanisms for transmitted data.

- *Automotive Company Applications' Center*. In the literature, the automotive company applications' center has different names, e.g. a *portal* or a *remote service center*. To summarize, it consists of a set of servers providing services to their vehicles. It holds necessary information about the vehicle, such as information from previous

services (e.g., diagnostics data), configuration data, cryptographic keys, as well as new software available for the ECUs.

- *Third Party Applications' Center*. Apart from services provided by the automotive company, third party services can be provided to the vehicle. We could imagine that large "application stores" for vehicles will be available in the future. These applications can provide any kind of service to the vehicle.

- *Trusted Network*. Some networks can be considered to be trusted by the applications' centers and the vehicle. For example, a repair shop may be considered to be a trusted network by the automotive company and the vehicle. In delivering a service to this network, it may well be that some requirements in an implementation can be relaxed.

  An example where the security requirements in the implemented service can be relaxed is when performing remote diagnostics over a wireless network in a repair shop. If appropriate link layer encryption is applied, the security of the wireless communication could be considered to be equal to that of a cable. This will not be the case when the communication with the vehicle is performed through the Untrusted Network; here end-to-end application encryption might be the only choice.

- *Untrusted Network*. All networks, except for the trusted networks, are considered to be untrusted. In these networks, the services provided to the vehicle have to be adapted to the hostile environment of the internet. In the same way as for the trusted networks, other local services may also be provided in these networks.

- *Internet Backbone*. The internet backbone, with its ISPs, is the core network for connecting the other four regions together. A backbone network is usually well protected and operated by network specialists in a networks operations center, NOC. Therefore, when network traffic has reached the internet backbone, it is assumed in the model that it is unlikely that the data will be intentionally modified.

12.1.2 Vehicle communication

The vehicle communication domain describes the possible communication means between the vehicle and the managed infrastructure and with other objects. The following communications scenarios exist.

- *Vehicle to Wireless AP*. The vehicle can establish a connection to a wireless AP. All open APs (hotspots) are considered to be part of the untrusted network. Furthermore, a protected AP, where the vehicle needs authentication keys, can be available in both trusted networks and in untrusted networks.

- *Vehicle to RSUs*. RSUs can also be used to establish a connection from the vehicle to other networks in the managed infrastructure.

- *Vehicle to Cellular Base Stations*. A mobile data network, can be used to establish a connection from the vehicle to the internet.
- *Vehicle to Mobile Devices*. Mobile devices can be connected to the vehicle. For example, a connection can be established to a mobile phone, a laptop, or a PDA. Furthermore, the vehicle can also act as a gateway for the mobile device, so that the mobile device can reach the same network as the vehicle.
- *Vehicle to Cellular Base Station via a Mobile Device*. If the vehicle lacks the possibility to connect directly to a cellular base station, another mobile device with a connection to the cellular base station can be used as a gateway. One example is to use the driver's mobile phone.
- *Vehicle to other Vehicles*. Finally, the vehicle can connect to other vehicles and create a VANET. This V2V communication will be critical in future traffic- and safety-related services.

It should be noted that the description of the vehicle communication above is based on just one vehicle; any connected car will have the same communication surroundings. This means that the vehicle may possibly reach the managed infrastructure, via other vehicles or other mobile devices acting as gateways.

12.1.3 Using the framework to assess the security of vehicle services

From the model of the infrastructure of the connected car, there are different aspects that can be discussed regarding the V2V and the V2I communication. One of them is the security of the services delivered to the vehicle. Figure 2.1 presents a brief taxonomy of the security of these services. Four categories are described: the *actors*, the *V2X communication technologies*, *network paths*, and the *dependability and security attributes*. A description of them follows below.

- *Actors*. Six different actors that can be involved in a service have been identified. Common for them all are that they have interests in how the service is being designed and delivered; the automotive company and the application provider can state requirements, the car owner and the driver can have concerns on how the data from a service is processed, the authorities can issue legal requirements, and an attacker can try to manipulate the service in an unwanted way.
- *V2X Communication Technologies*. A number of communication technologies are available for connecting the vehicle to other devices. Examples of these are listed. An extended list, including classifications of the communication technologies, can be found in [247].
- *Network Paths*. The service may be delivered to the vehicle using one of several network paths. The model describes four possible network paths that the service

can be delivered through (see Figure 12.1): the trusted network, the untrusted network, the internet backbone, and an ad-hoc network.

- *Dependability and Security Attributes*. To deliver the service in a secure and safe manner, the six attributes for dependability and security need to be considered [248].

From these four categories, an analysis can be made to further clarify how a service will work in the infrastructure, and also highlight the dependability and security attributes that need to be addressed in providing such a service.

With the security assessment tree, the problem with the vast number of issues that need to be considered in securing a service, is identified. It helps us to state requirements regarding security and provides us with a framework and a template for security assessment by identifying threats and communication patterns and to define countermeasures.

12.2 Certifications

Even if standards for security design are developed and legal requirements are defined, we still face the problem of *guaranteeing* that a particular design fulfils all requirements. Certification of individual components and functions using procedures such as *Common Criteria* (CC) and *FIPS* may be a way to, at least guarantee a sound design. Such certifications may be useful for individual components in a vehicle, but will not be applicable to a whole network in a car since any change of the software or hardware requires a full re-certification of all software in the vehicle, a very time consuming and work-intensive process.

It is reasonable to assume that all V2X traffic is sent by stations similar to ETSI's ITS station that fulfil a wide range of security requirements, and such components used by many car manufacturers are good targets for certification. The use of standardized and certified communication nodes for V2X short range communications have several advantages, not just for interoperability, but cooperative development of common modules can enhance security significantly and enable certification to be done.

Our conclusion is therefore that certifications are to some degree useful but come with several shortcomings: First of all, only smaller components can be certified to higher levels, otherwise the increased complexity makes it impossible to prove any functionality. Therefore, only individual components, such as firewall functionality in a subsystem can be certified, but not a full vehicle with hundreds of communicating ECUs. Second, to require only certified products in vehicles may prevent deployment of new functionality and create unacceptable delays of patches to known problems.

12.2.1 Common Criteria Certification

Common Criteria certification is an ISO standard often used when security devices are evaluated. It uses different *Evaluation Assurance Levels* (EAL1 to EAL7) with different requirements on the *target of evaluation*, TOE. The higher the level is, the higher are

the requirements. The targets are evaluated according to some criteria, *claims*, often specified in a *Security Target* written by the organization that wants to certify a device. It is not possible to compare two devices even if they both have been certified at the same level if the evaluation criteria differ. Therefore, for some types of devices such as firewalls and operating systems, there exist a pre-defined sets of criteria they should be validated against, called *Protection Profiles* to enable such a comparison. It also means that with weak validation criteria, it is possible to achieve a high assurance level but not necessary having a more secure device. A good example of this is that Microsoft Windows XP was certified at Level 4 (EAL 4) which is probably as high as a complex system like an operating system will reach, but it is still not considered to be a secure platform at all. Therefore, by relaxing the the criteria to evaluate against, it is possible to reach a high level in the certification.

The larger and more complex a system is, the harder it becomes to certify. Levels EAL1 to EAL4 are possible to reach with targets like conventional operating systems and products based on (executing in) such systems. Up to this level, it is enough to be able to motivate that it is reasonable to assume that the target of evaluation can fulfill the criteria and have a reasonably sound design, since level EAL4 states that the target should be methodically designed, tested, and reviewed. This is mostly done through documentation which states how and by what modules in the code the functions are implemented. Higher levels (EAL5-EAL6) require semi-formal design and verification which soon becomes too complex for larger targets, and EAL7 requires formally verified design and verification.

Even if it would be possible to certify the functionality of a full vehicle at level EAL4, a problem still exists when new functionality is added or patches are applied. A re-certification is needed as soon as *any* functionality or code is changed. Delta certifications are permitted where only the changes and documentation that shows that the changes do not affect other security-sensitive functions, are re-evaluated, but a complete re-evaluation is needed on regular basis. This makes it is more or less impossible (or at least extremely costly and time consuming) to certify a full vehicle with all its components at any higher level and maintain that certification level over time. Another problem is that the a high certification of a complex target takes time and will significantly affect time to market in a negative way.

Our conclusion is therefore that certification of smaller components or devices within the vehicle make sense. Such devices could be TPM modules or isolated components that perform, for example, V2X communications. Preferably these devices are used by many car manufacturers which motivates the time, effort and cost to perform and maintain the certification. Work is also needed to create proper protection profiles that are useful for all security sensitive devices used in the vehicular domain.

Chapter 13

Standardization Organizations

In this Chapter, a survey of the most important standardization efforts that are related to security in the vehicular environment is shown. Their goal is to set the requirements, guidelines, and to standardize communication systems and platforms.

Standardization and provision of standardized communication between vehicles (V2V) and between vehicles and infrastructure (V2I) is done by various standardization organizations. The following bodies are responsible for planning, development and adoption of the European standards:

13.1 Institute of Electrical and Electronics Engineers (IEEE)

A non-profit organization based in United States with members in about 160 different countries mainly focused on publications, educational activities, standards, and development processes. The IEEE 1609 which is a set of standards for Wireless Access in Vehicular Environments (WAVE) has been defined, standardized and maintained by the IEEE 1609 Working Group. The IEEE 1609 WAVE standard defines a communication stack and a set of services and security mechanisms that enable secure V2V and V2I wireless communications. These communications are based on the IEEE 80211.p link-layer protocol. To ensure the compatibility with US antenna and radio transceivers, the IEEE 802.11p is also used in Europe as a basis for ITS G5 standard defined by ETSI. According to the IEEE Standards Association [39] the existing standards listed in Table 13.1 are created by the IEEE 1609 WAVE Working Group.

13.2 International Organization for Standardization (ISO)

ISO/TC 204, ITS, was created in 1992 and all its ITS activities are organized in 14 working groups. The exception is the in-vehicle transport information and control systems area which is covered by TC 22. They are closely cooperating with ETSI TC ITS, see below.

The *ISO 26262* standard [114], which is an adaptation of the functional safety IEC 61508 standard, provides methods to mitigate the effect of faults and random failures in

Active Standards	
IEEE 1609.3-2016	WAVE-Networking Services
IEEE 1609.4-2016	WAVE-Multi-channel Operation
IEEE 1609.11-2010	WAVE-Over-the-Air Electronic Payment Data Exchange Protocol for ITS
IEEE 1609.2-2016	WAVE-Security Services for Applications and Management Messages
IEEE 1609.12-2016	WAVE-Identifier Allocations
Withdrawn Standards	
IEEE 1609.1-2006	Trial-Use Standard for WAVE - Resource Manager
IEEE 1609.2-2006	Trial-Use Standard for WAVE - Security Services for Applications and Management Messages
Superseded Standards	
IEEE 1609.3-2007	Trial-Use Standard for WAVE - Networking Services

Table 13.1: Standards created by the IEEE 1609 WAVE Working Group

hardware. Automotive Safety-integrity Level (ASIL), defined in ISO 26262, is a way to certify components in the automotive industry with respect to acceptable failure rates and can be used to control and predict the failure behavior of components. The intention is to assess and be aware of the *impact* and possible *damage* that may emerge from failures of components in the vehicle. The ASIL standard addresses safety only (not security) and introduces four different safety levels (1-4), the highest (4) required by safety critical components.

ASIL can be useful when defining requirements and evaluating particular components (ECUs), but when components communicate and use data from other components, it becomes hard to foresee all possible failure modes of the whole system. And even if security is not addressed today, it will affect how critical components in the system can be designed.

13.3 International Electrotechnical Commission (IEC)

The International Electrotechnical Commission (IEC) [249] is a not-for-profit and non-governmental organization, founded in 1906, which develops international standards and operates conformity assessment systems in the fields of electrotechnology.

The IEC comprises one member National Committee per country, they each pay membership fees and in exchange can participate fully in IEC work. The most relevant standard in the automotive context is IEC 61508, which is about functional safety.

13.4 European Telecommunications Standards Institute (ETSI)

ETSI has received the mandate to standardize V2V communications within the European Union. Although ETSI has as its primary responsibility standardization work in the telecommunications sector, it also has a committee working with ITS deployment dealing with applications, security and networking.

They have standardized an *ITS station* which is intended to be used in all external vehicular communications, both in vehicles and in road side units [41]. The standard describes the functionality that should be contained in all nodes participating in V2X communications. ETSI has also published a Threat, Vulnerability and Risk Analysis (TVRA) methodology which is applied to the proposed ITS station with its communication and a design of the needed security services [5, 250]. This methodology is supposed to be used to assess and evaluate security related functionality in the vehicular domain.

The ETSI ITS Technical Committee (TC) has different working groups: WG1 develops the basic set of application requirements and services, WG2 provides the architecture specification and addresses the cross layer issues, WG3 provides the 5.9 GHz network and transport protocols, WG4 provides the European profile investigation of 802.11p, and WG5 works with the security architecture

More about the ETSI project and the ITS station is discussed in Section 6.2.

13.5 European Committee for Standardization (CEN)

European Committee for Standardization (CEN) is an international non-profit association created in 1975. It is a major provider of European Standards and technical specifications. Most of the activities in ITS are developed within the CEN/TC 278 “Road transport and traffic telematics”. The technical committee has several working groups working on DSRC, eSafety and Co-operative systems.

13.6 European Council for Automotive R&D (EUCAR)

The European Council for Automotive R&D (EUCAR) [251] is the European body for collaborative automotive and road transport R&D. EUCAR is an industrial association owned by its members, which are the 14 major European manufacturers of cars, trucks and busses.

13.7 Car2Car Communication Consortium (C2C-CC)

Car 2 Car Communication Consortium (C2C-CC) is an industry consortium initiated by European car manufacturers in summer 2002. It is an open non-profit organization with several partners and mainly consists of research institutes, car manufacturers and

suppliers. C2C-CC cooperates closely with ETSI TC ITS and the ISO/TC 204 on the specification of the ITS European and ISO standards.

The main goal of the consortium is standardization of protocols and interfaces used in wireless communication between vehicles and infrastructure, i.e. most V2X communications. The aim is to make the different vehicle brands and road-side objects interoperable.

They have also developed a reference architecture which can be used to assess security in communicating vehicles, see Section 12.1

13.8 National Highway Traffic Safety Administration (NHTSA)

The National Highway Traffic Safety Administration (NHTSA) [252], under the U.S. Department of Transportation, deals with traffic safety in the US. While US standards are of course not directly applicable to the EU, there is a regular exchange of ideas, and standards in the US and the EU often influence each other.

NHTSA is responsible for reducing deaths, injuries and economic losses resulting from motor vehicle crashes. This is accomplished by setting and enforcing safety performance standards for motor vehicles and motor vehicle equipment, and through grants to state and local governments to enable them to conduct effective local highway safety programs.

13.9 AUTOSAR Development Partnership

AUTOSAR [253] is a consortium of automotive companies and suppliers that have an interest in an open software platform for ECUs. AUTOSAR is essentially a specification for an operating system, but the actual implementation depends on the concrete supplier.

The main goals of the project are to allow 3rd party software development and to increase software interoperability in the automotive industry. AUTOSAR describes in [113] the basic security features and their functionality.

13.10 SAE International

SAE International started out as the U.S.-based Society of Automotive Engineers. Nowadays it is an international organization for the professional transport industry and is involved with developing standards and identifying best practices in the industry.

Chapter 14

Research Projects

The European Commission research and innovation programs fund several framework programs with a variety of topics. Horizon 2020 is the currently ongoing program where the majority of the R&D activities within ITS are handled. Projects from the 7^{th} Framework Program (2007–2013) are ongoing or already finished. Some of the more influential research projects within FP6 and FP7 are introduced here. ITS related projects of Horizon 2020 are listed in Section 14.6.

The mentioned projects are almost all geographically located in Europe, and their outcome mainly affect ETSI (European Telecommunications Standards Institute) and therefore also end up in ISO for standardization.

14.1 CARONTE (Creating an Agenda for Research on Transportation sEcurity)

Creating an Agenda for Research on Transportation sEcurity (CARONTE) [254] was a European project from 2014 - 2016. The project's aim is to create a strategic research plan for security in land transport. The results of this project are the identification and classification of research in this particular area. The identified top priorities for land transportation security are [254]:

- staying operational in the event of a cyber-incident,
- timely and efficient threat detection, and
- special security problems of railways as open systems (an integrated approach).

The results highlight the necessity of a secure system even though a cyber-incident occured. The proposed research title for this topic is "An integrated approach to assure cyber resilience in land transportation".

14.2 HEAVENS (HEAling Vulnerabilities to ENhance Software Security and Safety)

The HEAVENS project (2013 - 2016) can be seen as the predecessor of the HOLIstic Approach to Improve Data SECurity (HoliSec) project. The project's findings were the introduction of basic dependability and security concepts, basic concepts of system engineering processes from security perspective, baseline architectures of ITS, in-vehicle network and in-vehicle software to be used in the HEAVENS project, and the identification, explanation of use cases that lead to a better understanding of security needs in the automotive E/E systems, and the security requirements based on the needs derived from the use cases [138].

The HEAVENS security model has also received much attention from standardization organizations, such as SAE J3061 [115], AUTOSAR WP-X-SEC and NHTSA. This model is a systematic approach that includes methods, processes and tool support to derive security requirements and to perform security testing and evaluation for automotive E/E systems.

14.3 SeFram

SeFram, 2012-2015, was a Swedish research project in which Volvo Cars Corporation and Chalmers are the main partners. The goals of the project were:

- securing communication within the car and between the car and the outside infrastructure,
- developing models and define security requirements for vehicle communication, and
- creating a practically useful framework for future security work within the connected car.

The project is a continuation of earlier projects in this area (SIGYN I and II). The project has resulted in an in-depth analysis of how to offer secure remote diagnostics for vehicles, and especially securing the repair shop and its communications. A new protocol using a trusted third party has been developed with the intention to protect vehicles against unauthorized access from compromised diagnostics units. The protocol has been formally verified to be correct and able to guarantee some characteristics. Work has also been done in protecting the repair shop network from unauthorized access by vehicles, i.e. from access by vehicles not scheduled for service or remote diagnostics.

Another task being performed within this project was to create a test implementation of the ETSI ITS station, see Section 13.4. The implementation has revealed several problems with the standard and weaknesses in the description of essential parts. The revealed problems are described in [255] and lead to a modification of the ETSI ITS specification.

14.4 SESAMO (Security and Safety Modelling)

SESAMO [256], 2012-2015, is a European project that addresses the problems arising with convergence of safety and security in embedded systems at architectural level, where subtle and poorly understood interactions between functional safety and security mechanisms impede system definition, development, certification, and accreditation procedures and standards.

14.5 EVITA (E-safety Vehicle Intrusion protected Applications)

EVITA [4] was a European project, funded under the 7th Framework Program (FP7) 2008-2011. Its main objective was to design, verify and implement a hardware security module to be used in an architecture for securing on-board networks. In this architecture, security relevant components are protected against tampering and sensitive data are protected against compromise. By focusing on protecting the intra-vehicle communication, EVITA complements other projects which mainly focus on external (V2X) communication.

In EVITA, an architecture is created which should enable ECUs to implement cryptography operations in a secure manner. The ECU is equipped with a cryptographic co-processor protected in a HSM. This module is responsible for performing all cryptography applications. More details on the HSM can be found in Section 8.1.4.

14.6 European Projects - Horizon 2020

Several projects funded under Horizon 2020 are about securing Cyber Physical System (CPS). The automotive sector is thereby considered in many projects as a use case or as area of interest. Table 14.1 lists ongoing related projects in course of Horizon 2020.

Table 14.1: European Projects in the course of Horizon 2020 related to the automotive sector

Project Name	**Description**
SAFURE [257] 2015 – 2018	*SAFety and secURity by design for interconnected mixed-critical cyber-physical systems:* The automotive sector is mentioned as one of three use cases for the proof-of-concept. The results will be a framework with the capability to detect, prevent and protect from security threats on safety, able to monitor from application level down to the hardware level potential attacks to system integrity from time, energy, temperature and data threats; methodology that supports design of safety and security of embedded systems (inlc. tools and modelling language extensions); proof-of concept through three use cases; specifications to design and develop SAFURE compliant products.

SafeCOP [258] 2016 – 2019	*Safe Cooperating Cyber-Physical Systems using Wireless Communication:* SafeCOP will establish a safety assurance approach, a platform architecture, and tools for cost-efficient and practical certification of cooperating CPS (CO-CPS). SafeCOP defines a platform architecture and develops methods and tools, which will be used to produce safetey assurance evidence needed to certify cooperative functions. SafeCOP will extend current wireless technologies to ensure safe and secure cooperation. The project contributes to new standards and regulations by providing scientifically validated solutions.
SCOUT [259] 2016 – 2019	*Safe and COnnected aUtomation in road Transport:* Identifies pathways for an accelerated spread of safe and connected high-degree automated driving in Europe. The project considers: the user needs and expectations, technical and non-technical gaps and risks, and viable business models. It gathers use cases for connected automation in road transport, assesses and ranks the non-technical and technical gaps, and risks for the implementation of these use cases. Sustainable business models will be also identified. Furthermore, advice for policies and regulatory frameworks for safe and connected automation will then be derived on the basis of the aforementioned analysis with the support of a network of relevant stakeholders, and classified into a common roadmap aiming to implement the formulated vision for "Safe and connected automation in 2030".
SHARCS [260] 2015 – 2018	*Secure Hardware-Software Architectures for Robust Computing Systems:* A framework for designing, building and demonstrating secure-by-design applications and services, that achieve end-to-end security for their users. SHARCS will achieve this by systematically analyzing and extending, as necessary, the hardware and software layers in a computing system. The framework will be assessed by using a diverse set of security-critical, real-world applications (from different domains: medical, cloud and automotive).
CYRail [261] 2016 – 2018	*Cybersecurity in the RAILway sector:* This project is concerned with the security of CPS in the area of railways. It aims to deliver tailored specifications and recommendations for secure modern rail systems design and operation. The challenges are related to the automotive industry: wide and distributed geographical display limit the traditional cyber-protection and cyber-defence tools & practices, heterogeneous nature of rail systems attacks; passenger-centric services may expose rail systems to threats.

IMMORTAL [262] 2015 – 2018	*Integrated Modelling, Fault Management, Verification and Reliable Design Environment for Cyber-Physical Systems:* Development of an integrated, cross-layer modelling based tool framework for fault management, verification and reliable design of dependable CPS.

Chapter 15

Conclusions

This deliverable consists of a discussion about several demonstrated security threats and their implications. It is shown why security is important in the vehicular domain. How the different security aspects have been addressed by various research projects is described in this document as well. We also commented on the practicability of proposed solutions. The report covers security in the internal (in-vehicle) as well as in the external (V2X) communication.

Securing the in-vehicle communication with traditional mechanisms has been discussed. These mechanisms are among others internal separation of traffic, the use of message authentication codes (MACs) to guarantee message integrity, firewalls for both external traffic and for internal traffic implemented in gateway ECUs, use of IDSs, use of certificates for identification and verification of devices or software (vehicles, road-side objects, ECUs, drivers, and firmware), and the problems with distributing revocation lists (CRLs). Tamper-proof hardware security modules (HSM modules) to speed-up cryptographic operations and to offer a safe storage for private or symmetric keys have been discussed as well.

The goal of this document was to write an objective and comprehensive summary of this interesting field. We hope that the work will lead to an increased understanding and be an inspiration to future work to make road traffic even safer and more secure than it is today.

New vehicle applications will soon be introduced that communicate with applications in other vehicles and road-side units (V2X). Examples are applications that process information messages from traffic lights, road signs and other vehilces in urban traffic. Other applications will be offered by car manufacturers, such as remote software updates, remote diagnostics and other services for car owners and drivers. Other applications will be offered by third parties, for example applications for automatic road toll payments, navigational systems and automatically transmitted vehicle reports about current road conditions. Security work must address how to implement these applications in a secure and safe way, how to isolate safety critical functions from all “nice to have” applications

and implement protection against attacks, internal and external, that may compromise the vehicle. In short, we must find ways to guarantee the safety of the vehicle.

Standardization work has begun although the work has mainly focused on low-level communications and protocols. Security design and architecture has only been addressed to some degree, for example in the ITS station standardization work done by ETSI. But it still remains to be seen if the proposed standards will be universally accepted by the industry and all countries in the world. We provided an overview over current standardization efforts and what they provide us with respect to security.

Internal security is more or less absent in vehicles. In this report, we also discuss how to prevent that any device with access to the internal bus can send falsified messages.

Bibliography

[1] K. Koscher, A. Czeskis, F. Roesner, S. Patel, T. Kohno, S. Checkoway et al. 'Experimental Security Analysis of a Modern Automobile'. In: *2010 IEEE Symposium on Security and Privacy (SP)*. IEEE, 2010, pp. 447–462. ISBN: 1424468949. DOI: 10.1109/SP.2010.34 (cit. on pp. 1, 15, 54, 55, 57).

[2] QualComm. *eCall Whitepaper, v1.5*. QualComm. Mar. 2009. URL: http://ec.europa.eu/information_society/activities/esafety/doc/ecall/pos_papers_impact_assessm/qualcomm.pdf (cit. on p. 3).

[3] National Highway Traffic Safety Administration (NHTSA). *U.S. DOT advances deployment of Connected Vehicle Technology to prevent hundreds of thousands of crashes*. URL: https://www.nhtsa.gov/press-releases/us-dot-advances-deployment-connected-vehicle-technology-prevent-hundreds-thousands (visited on 19th Dec. 2016) (cit. on p. 4).

[4] EVITA Project. *E-safety Vehicle Intrusion Protected Applications (EVITA)*. URL: http://www.evita-project.org/ (visited on 14th Nov. 2016) (cit. on pp. 7, 10, 55, 75, 113).

[5] ETSI. *Intelligent Transport Systems (ITS); Security; Threat, Vulnerability and Risk Analysis (TVRA)*. Technical Report TR 102 893, v1.1.1. 650 Route des Lucioles, F-06921 Sophia Antipolis Cedex, France: ETSI, Mar. 2010 (cit. on pp. 7, 28, 109).

[6] P. Kleberger, A. Javaheri, T. Olovsson and E. Jonsson. 'A Framework for Assessing the Security of the Connected Car Infrastructure'. In: *Proceedings of the Sixth International Conference on Systems and Networks Communications (ICSNC 2011)*. Barcelona, Spain, Oct. 2011, pp. 236–241. ISBN: 978-1-61208-166-3. URL: http://www.thinkmind.org/index.php?view=article&articleid=icsnc_2011_10_30_20229 (cit. on pp. 8, 72, 74, 101).

[7] SysSec. *D6.1: Report on the State of the Art of Securit in Sensor Networks*. URL: http://www.syssec-project.eu/m/page-media/3/syssec-d6.1-SoA-SecurityInSensorNetworks.pdf (visited on 30th Nov. 2013) (cit. on p. 11).

[8] H. Lee, H.-M. Tsai and O. Tonguz. 'On the Security of Intra-Car Wireless Sensor Networks'. In: *Vehicular Technology Conference Fall (VTC 2009-Fall), 2009 IEEE 70th*. 2009, pp. 1–5. DOI: 10.1109/VETECF.2009.5378964 (cit. on p. 11).

[9] B. Preneel. 'The First 30 Years of Cryptographic Hash Functions and the NIST SHA-3 Competition'. In: *Topics in Cryptology - CT-RSA 2010*. Ed. by J. Pieprzyk. Lecture Notes in Computer Science 5985. Springer Berlin Heidelberg, Jan. 2010, pp. 1–14. ISBN: 978-3-642-11924-8, 978-3-642-11925-5. URL: http://link.springer.com/chapter/10.1007/978-3-642-11925-5_1 (visited on 28th Nov. 2013) (cit. on p. 12).

[10] S. Checkoway, D. McCoy, B. Kantor, D. Anderson, H. Shacham, S. Savage, K. Koscher, A. Czeskis, F. Roesner and T. Kohno. 'Comprehensive Experimental Analyses of Automotive Attack Surfaces'. In: *Proceedings of the 20th USENIX Security Symposium*. San Francisco, CA, USA, Aug. 2011, pp. 77–92 (cit. on p. 16).

[11] I. Rouf, R. Miller, H. Mustafa, T. Taylor, S. Oh, W. Xu, M. Gruteser, W. Trappe and I. Seskar. 'Security and Privacy Vulnerabilities of In-Car Wireless Networks: A Tire Pressure Monitoring System Case Study'. In: *Proceedings of the 19th USENIX conference on Security*. USENIX Security'10. Washington, DC: USENIX Association, 2010, pp. 21–21. ISBN: 888-7-6666-5555-4. URL: http://dl.acm.org/citation.cfm?id=1929820.1929848 (cit. on p. 16).

[12] T. Fox-Brewster and Forbes. *BMW Update Kills Bug In 2.2 Million Cars That Left Doors Wide Open To Hackers*. Feb. 2015. URL: http://www.forbes.com/sites/thomasbrewster/2015/02/02/bmw-door-hacking (visited on 12th Dec. 2016) (cit. on p. 17).

[13] A. Greenberg and Forbes. *Hackers Reveal Nasty New Car Attacks–With Me Behind The Wheel*. Aug. 2013. URL: http://www.forbes.com/sites/andygreenberg/2013/07/24/hackers-reveal-nasty-new-car-attacks-with-me-behind-the-wheel-video/#1990f3e85bf2 (visited on 8th Dec. 2016) (cit. on p. 18).

[14] C. Miller and C. Valasek. 'Remote Exploitation of an Unaltered Passenger Vehicle'. In: *Blackhat USA* 2015 (2015), pp. 1–91. URL: http://illmatics.com/Remote%20Car%20Hacking.pdf (cit. on pp. 18, 19).

[15] wired.com. *The Jeep Hackers Are Back to Prove Car Hacking Can Get Much Worse*. URL: https://www.wired.com/2016/08/jeep-hackers-return-high-speed-steering-acceleration-hacks/ (visited on 8th Dec. 2016) (cit. on p. 19).

[16] Vulnerability Lab. *BMW ConnectedDrive - (Update) VIN Session Vulnerability*. July 2016. URL: https://www.vulnerability-lab.com/get_content.php?id=1736 (visited on 17th Nov. 2016) (cit. on p. 20).

[17] Vulnerability Lab. *BMW - (Token) Client Side Cross Site Scripting Vulnerability*. July 2016. URL: https://www.vulnerability-lab.com/get_content.php?id=1737 (visited on 17th Nov. 2016) (cit. on p. 20).

[18] Keen Security Lab of Tencent. *Car Hacking Research: Remote Attack Tesla Motors*. Sept. 2016. URL: http://keenlab.tencent.com/en/2016/09/19/Keen-Security-Lab-of-Tencent-Car-Hacking-Research-Remote-Attack-to-Tesla-Cars/ (visited on 17th Nov. 2016) (cit. on p. 20).

[19] Promon AS. *Updates and precisions on the Tesla hack*. Nov. 2016. URL: https://promon.co/blog/updates-precisions-tesla-hack/ (visited on 25th Nov. 2016) (cit. on p. 20).

[20] R. A. Posner. *Economic analysis of law*. English. 5. New York: Aspen Law & Business, 1998. ISBN: 1567065627;9781567065626; (cit. on p. 21).

[21] A. F. Westin. 'Privacy and freedom'. In: *Washington and Lee Law Review* 25.1 (1968), p. 166. URL: http://scholarlycommons.law.wlu.edu/cgi/viewcontent.cgi?article=3659&context=wlulr (visited on 12th Dec. 2016) (cit. on p. 21).

[22] N. Li, M. Lyu, D. Su and W. Yang. 'Differential Privacy: From Theory to Practice'. In: *Synthesis Lectures on Information Security, Privacy, and Trust* 8.4 (25th Oct. 2016), pp. 1–138. ISSN: 1945-9742. DOI: 10.2200/S00735ED1V01Y201609SPT018. URL: http://www.morganclaypool.com/doi/abs/10.2200/S00735ED1V01Y201609SPT018 (visited on 28th Nov. 2016) (cit. on p. 21).

[23] L. Sweeney. 'k-anonymity: A model for protecting privacy'. In: *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems* 10.5 (2002), pp. 557–570. URL: http://www.worldscientific.com/doi/abs/10.1142/S0218488502001648 (visited on 15th Dec. 2014) (cit. on p. 21).

[24] A. Narayanan and V. Shmatikov. 'Robust De-anonymization of Large Sparse Datasets'. In: *IEEE Symposium on Security and Privacy, 2008. SP 2008*. IEEE Symposium on Security and Privacy, 2008. SP 2008. May 2008, pp. 111–125. DOI: 10.1109/SP.2008.33 (cit. on p. 22).

[25] M. Barbaro and T. Zeller. 'A Face Is Exposed for AOL Searcher No. 4417749'. In: *The New York Times* (9th Aug. 2006). ISSN: 0362-4331. URL: http://query.nytimes.com/gst/abstract.html?res=9E0CE3DD1F3FF93AA3575BC0A9609C8B63 (visited on 16th July 2015) (cit. on p. 22).

[26] X. Gao, B. Firner, S. Sugrim, V. Kaiser-Pendergrast, Y. Yang and J. Lindqvist. 'Elastic Pathing: Your Speed is Enough to Track You'. In: *Proceedings of the 2014 ACM International Joint Conference on Pervasive and Ubiquitous Computing*. UbiComp '14. New York, NY, USA: ACM, 2014, pp. 975–986. ISBN: 978-1-4503-2968-2. DOI: 10.1145/2632048.2632077. URL: http://doi.acm.org/10.1145/2632048.2632077 (visited on 20th Sept. 2016) (cit. on p. 22).

[27] M. G. Lee, Y. K. Park, K. K. Jung and J. J. Yoo. 'Estimation of fuel consumption using in-vehicle parameters'. In: *International Journal of u-and e-Service, Science and Technology* 4.4 (2011), pp. 37–46. URL: http://www.sersc.org/journals/IJUNESST/vol4_no4/3.pdf (visited on 20th Sept. 2016) (cit. on p. 22).

[28] C. Dwork. 'Differential privacy'. In: *Automata, languages and programming*. Springer, 2006, pp. 1–12. URL: http://link.springer.com/chapter/10.1007/11787006_1 (visited on 5th Dec. 2014) (cit. on p. 22).

[29] P. Samarati and L. Sweeney. *Protecting privacy when disclosing information: k-anonymity and its enforcement through generalization and suppression*. Technical report, SRI International, 1998. URL: http://epic.org/privacy/reidentification/Samarati_Sweeney_paper.pdf (visited on 2nd Feb. 2015) (cit. on p. 23).

[30] A. Machanavajjhala, D. Kifer, J. Gehrke and M. Venkitasubramaniam. 'L-diversity: Privacy beyond k -anonymity'. In: *ACM Transactions on Knowledge Discovery from Data* 1.1 (1st Mar. 2007), 3–es. ISSN: 15564681. DOI: 10.1145/1217299.1217302. URL: http://portal.acm.org/citation.cfm?doid=1217299.1217302 (visited on 18th Dec. 2014) (cit. on p. 24).

[31] N. Li, T. Li and S. Venkatasubramanian. 't-Closeness: Privacy Beyond k-Anonymity and l-Diversity'. In: *2007 IEEE 23rd International Conference on Data Engineering*. Apr. 2007, pp. 106–115. DOI: 10.1109/ICDE.2007.367856 (cit. on p. 24).

[32] F. Kargl, P. Papadimitratos, L. Buttyan, M. Muter, E. Schoch, B. Wiedersheim et al. 'Secure Vehicular Communication Systems: Implementation, Performance, and Research Challenges'. In: *IEEE Communications Magazine* 46.11 (Nov. 2008), pp. 110–118. DOI: 10.1109/MCOM.2008.4689253 (cit. on pp. 24, 46, 62).

[33] M. Gerlach, A. Festag, T. Leinmüller, G. Goldacker and C. Harsch. 'Security Architecture for Vehicular Communication'. In: *4th International Workshop on Intelligent Transportation (WIT)*. Hamburg, Germany, Mar. 2007 (cit. on p. 25).

[34] European Parliament, Council of the European Union. 'Regulation (EU) 2016/679 of the European Parliament and of the Council of 27 April 2016 on the protection of natural persons with regard to the processing of personal data and on the free movement of such data, and repealing Directive 95/46/EC (General Data Protection Regulation)'. In: *Official Journal of the European Union* (4th May 2016). URL: http://eur-lex.europa.eu/legal-content/EN/TXT/PDF/?uri=CELEX:32016R0679&qid=1473158599287&from=EN (visited on 6th Sept. 2016) (cit. on p. 25).

[35] European Commission. *European Commission - PRESS RELEASES - Press release - Questions and Answers - Data protection reform*. European Commission - PRESS RELEASES - Press release - Questions and Answers - Data protection reform. 21st Dec. 2015. URL: http://europa.eu/rapid/press-release_MEMO-15-6385_en.htm (visited on 12th Dec. 2016) (cit. on p. 25).

[36] W. John and T. Olovsson. 'Detection of malicious traffic on back-bone links via packet header analysis'. In: *Campus-Wide Information Systems* 25.5 (2008), pp. 342–358. ISSN: 1065-0741. DOI: 10.1108/10650740810921484. URL: http://www.emeraldinsight.com/journals.htm?articleid=1753842&show=abstract (cit. on p. 27).

[37] P. Kleberger, T. Olovsson and E. Jonsson. 'Security Aspects of the In-Vehicle Network in the Connected Car'. In: *2011 IEEE Intelligent Vehicles Symposium (IV)*. Baden-Baden, Germany, June 2011, pp. 528–533. DOI: 10.1109/IVS.2011.5940525 (cit. on p. 27).

[38] I. Studnia, V. Nicomette, E. Alata, Y. Deswarte, M. Kaaniche and Y. Laarouchi. 'Survey on security threats and protection mechanisms in embedded automotive networks'. In: *2013 43rd Annual IEEE/IFIP Conference on Dependable Systems and Networks Workshop (DSN-W)*. 2013, pp. 1–12. DOI: 10.1109/DSNW.2013.6615528 (cit. on p. 27).

[39] IEEE. *1609 WG - Dedicated Short Range Communication Working Group*. Oct. 2013. URL: https://standards.ieee.org/develop/wg/1609_WG.html (cit. on pp. 27, 107).

[40] European Commission. *M/453 EN: Standardisation mandate addressed to CEN, CENELEC and ETSI in the field of Information and Communication Technologies to support the interoperability of Co-operative systems for Intelligent Transport in the European Community*. European Commission. Oct. 2009 (cit. on p. 28).

[41] ETSI. *Intelligent Transport Systems (ITS); Communications Architecture*. European Standard EN 302 665, v1.1.1. 650 Route des Lucioles, F-06921 Sophia Antipolis Cedex, France: ETSI, Sept. 2010 (cit. on pp. 28, 36, 109).

[42] ETSI. *Intelligent Transport Systems (ITS); Security; Security Services and Architecture*. Technical Specification TS 102 731, v1.1.1. 650 Route des Lucioles, F-06921 Sophia Antipolis Cedex, France: ETSI, Sept. 2010 (cit. on p. 28).

[43] J. Hoagland, O. Whitehouse, T. Newsham, M. Conover and O. Friedrichs. 'Vista's Network Attack Surface'. Presented at CanSecWest. Apr. 2007. URL: http://hoagland.org/presentations/CanSecWest07-Vista-Ntw-Attack-Surface.pdf (cit. on p. 29).

[44] S. Habib, C. Jacob and T. Olovsson. 'An Analysis of the Robustness and Stability of the Network Stack in Symbian-based Smartphones'. In: *Journal of Networks* 4.10 (2009), pp. 968–975. DOI: 10.4304/jnw.4.10.968-975. URL: http://ojs.academypublisher.com/index.php/jnw/article/view/0410968975 (cit. on p. 29).

[45] A. Lang, J. Dittmann, S. Kiltz and T. Hoppe. 'Future Perspectives: The Car and Its IP-Address — A Potential Safety and Security Risk Assessment'. In: *Proceedings of the 26th International Conference on Computer Safety, Reliability, and Security (SAFECOMP '07)*. SAFECOMP '07. Nuremberg, Germany, Sept. 2007, pp. 40–53 (cit. on pp. 29, 57).

[46] E. Hamida, H. Noura and W. Znaidi. 'Security of Cooperative Intelligent Transport Systems: Standards, Threats Analysis and Cryptographic Countermeasures'. In: *Electronics* 4.3 (2015), pp. 380–423. ISSN: 2079-9292. DOI: 10.3390/electronics4030380. URL: http://www.mdpi.com/2079-9292/4/3/380/ (cit. on pp. 36, 42, 43, 46).

[47] F. Dressler, F. Kargl, J. Ott, O. Tonguz and L. Wischhof. 'Research Challenges in Intervehicular Communication: Lessons of the 2010 Dagstuhl Seminar'. In: *IEEE Communications Magazine* 49.5 (May 2011), pp. 158–164. ISSN: 0163-6804. DOI: 10.1109/MCOM.2011.5762813 (cit. on p. 37).

[48] M. Zhao, J. Walker and C.-C. Wang. 'Challenges and Opportunities for Securing Intelligent Transportation System'. In: *IEEE Journal on Emerging and Selected Topics in Circuits and Systems* 3.1 (2013), pp. 96–105. ISSN: 2156-3357. DOI: 10.1109/JETCAS.2013.2243633 (cit. on p. 37).

[49] D. Westhoff, B. Lamparter, C. Paar and A. Weimerskirch. 'On digital signatures in ad hoc networks'. In: *European transactions on telecommunications* 16.5 (2005), pp. 411–425 (cit. on p. 40).

[50] G. Calandriello, P. Papadimitratos, J.-P. Hubaux and A. Lioy. 'Efficient and robust pseudonymous authentication in VANET'. In: *Proceedings of the fourth ACM international workshop on Vehicular ad hoc networks*. VANET '07. Montreéal, Québec, Canada, Sept. 2007, pp. 19–28 (cit. on p. 40).

[51] ETSI. *Intelligent Transport Systems (ITS); Security; Trust and Privacy Management*. TS Technical Specification 102 941 V1.1.1. 2012 (cit. on pp. 40, 41).

[52] ETSI. *Intelligent Transport Systems (ITS);Security;ITS communications security architecture and security management*. TS Technical Specification 102 940 V1.1.1. 2012 (cit. on p. 41).

[53] ETSI (European Telecommunications Standards Institute). *ETSI TS 103 097 V1.2.1 (2015-06)-Intelligent Transport Systems (ITS); Security; Security header and certificate formats*. Tech. rep. 2015, pp. 1–33 (cit. on pp. 42, 43).

[54] M. Ring, T. Rensen and R. Kriesten. 'Evaluation of Vehicle Diagnostics Security–Implementation of a Reproducible Security Access'. In: *SECURWARE 2014* (2014), p. 213 (cit. on p. 45).

[55] M. Jenkins and S. M. Mahmud. 'Security Needs for the Future Intelligent Vehicles'. In: *2006 SAE World Congress*. Detroit, Michigan, USA, Apr. 2006 (cit. on p. 45).

[56] S. M. Mahmud, S. Shanker and I. Hossain. 'Secure Software Upload in an Intelligent Vehicle via Wireless Communication Links'. In: *IEEE Intelligent Vehicles Symposium, 2005. Proceedings*. 2005, pp. 588–593. DOI: 10.1109/IVS.2005.1505167 (cit. on p. 47).

[57] I. Hossain and S. M. Mahmud. 'Analysis of a Secure Software Upload Technique in Advanced Vehicles using Wireless Links'. In: *Proceedings of the IEEE Intelligent Transportation Systems Conference (ITSC 2007)*. 2007, pp. 1010–1015. DOI: 10.1109/ITSC.2007.4357797 (cit. on p. 47).

[58] I. Hossain and S. M. Mahmud. 'Secure Multicast Protocol for Remote Software Upload in Intelligent Vehicles'. In: *Proc. of the 5th Ann. Intel. Vehicle Systems Symp. of National Defense Industries Association (NDIA)*. Traverse City, Michigan, June 2005, pp. 145–155 (cit. on p. 47).

[59] D. K. Nilsson and U. E. Larson. 'Secure Firmware Updates over the Air in Intelligent Vehicles'. In: *IEEE International Conference on Communications Workshops, 2008. ICC Workshops '08*. May 2008, pp. 380–384. DOI: 10.1109/ICCW.2008.78 (cit. on p. 47).

[60] M. S. Idrees, H. Schweppe, Y. Roudier, M. Wolf, D. Scheuermann and O. Henniger. 'Secure Automotive On-Board Protocols: A Case of Over-the-Air Firmware Updates'. In: *Communication Technologies for Vehicles*. Ed. by T. Strang, A. Festag, A. Vinel, R. Mehmood, C. R. Garcia and M. Röckl. Lecture Notes in Computer Science 6596. Springer Berlin Heidelberg, Jan. 2011, pp. 224–238. ISBN: 978-3-642-19785-7, 978-3-642-19786-4. URL: http://link.springer.com/chapter/10.1007/978-3-642-19786-4_20 (visited on 19th Dec. 2016) (cit. on p. 48).

[61] *ISO/DIS 13400-1: Road vehicles — Diagnostic communication over Internet Protocol (DoIP) — Part 1: General information and use case definition*. Standard. International Organization for Standardization (cit. on p. 48).

[62] M. Johanson, P. Dahle and A. Söderberg. 'Remote Vehicle Diagnostics over the Internet using the DoIP Protocol'. In: *Proceedings of the Sixth International Conference on Systems and Networks Communications (ICSNC 2011)*. Barcelona, Spain, Oct. 2011, pp. 226–231. ISBN: 978-1-61208-166-3. URL: http://www.thinkmind.org/index.php?view=article&articleid=icsnc_2011_10_10_20096 (cit. on p. 48).

[63] D. Nilsson, L. Sun and T. Nakajima. 'A Framework for Self-Verification of Firmware Updates over the Air in Vehicle ECUs'. In: *2008 IEEE GLOBECOM Workshops*. Nov. 2008, pp. 1–5. DOI: 10.1109/GLOCOMW.2008.ECP.56 (cit. on p. 48).

[64] A. Weimerskirch. 'Secure Software Flashing'. In: *SAE Int. J. Passeng. Cars - Electron. Electr. Syst.* Vol. 2. 2009, pp. 83–86 (cit. on p. 48).

[65] A. Adelsbach, U. Huber and A.-R. Sadeghi. 'Secure Software Delivery and Installation in Embedded Systems'. In: *Embedded Security in Cars*. Ed. by K. Lemke, C. Paar and M. Wolf. 10.1007/3-540-28428-1_3. Springer Berlin Heidelberg, 2006, pp. 27–49. ISBN: 978-3-540-28428-4. URL: http://dx.doi.org/10.1007/3-540-28428-1_3 (cit. on p. 48).

[66] M.-j. Kang and J.-w. Kang. 'A Novel Intrusion Detection Method Using Deep Neural Network for In-Vehicle Network Security'. In: *PLoS ONE* 11.6 (2016), pp. 1–17. ISSN: 1932-6203. DOI: 10.1371/journal.pone.0155781 (cit. on pp. 50, 60).

[67] MOST Cooperation. 'MOST Specification Rev. 3.0 E2'. In: (2010), p. 32. URL: http://www.mostcooperation.com/ (cit. on p. 51).

[68] D. J. Gerhardt. 'Serial Data Communications Between Microprocessor Systems'. In: *SAE Technical Paper*. SAE International, Apr. 1986. DOI: 10.4271/860728. URL: http://dx.doi.org/10.4271/860728 (cit. on p. 51).

[69] SAE International. 'Serial Control and Communications Heavy Duty Vehicle Network - Top Level Document Superseding J1939 JUN2012'. In: SAE International, Aug. 2013. URL: https://saemobilus.sae.org/content/J1939_201308 (cit. on p. 52).

[70] M. Junger and Vector Informatik GmbH. *Introduction to J1939 - Application Note AN-ION-1-3100*. Apr. 2010. URL: https://vector.com/portal/medien/cmc/application_notes/AN-ION-1-3100_Introduction_to_J1939.pdf (visited on 8th Dec. 2016) (cit. on p. 52).

[71] Copperhill technologies. *A Brief Introduction to the SAE J1939 Protocol*. 2016. URL: http://copperhilltech.com/a-brief-introduction-to-the-sae-j1939-protocol/ (visited on 9th Dec. 2016) (cit. on p. 52).

[72] 'Automotive Ethernet: In-vehicle Networking and Smart Mobility'. In: *Design, Automation & Test in Europe Conference & Exhibition (DATE), 2013* (2013), pp. 1735–1739. ISSN: 15301591. DOI: 10.7873/DATE.2013.349. URL: http://ieeexplore.ieee.org/xpl/articleDetails.jsp?arnumber=6513795 (cit. on pp. 52, 53).

[73] L. L. Bello. 'Novel trends in automotive networks: A perspective on Ethernet and the IEEE Audio Video Bridging'. In: *Proceedings of the 2014 IEEE Emerging Technology and Factory Automation (ETFA)*. Sept. 2014, pp. 1–8. DOI: 10.1109/ETFA.2014.7005251 (cit. on p. 53).

[74] IEEE-Standards Association. *802.1BA-2011 - IEEE Standard for Local and metropolitan area networks–Audio Video Bridging (AVB) Systems*. Tech. rep. 2011 (cit. on p. 53).

[75] IEEE-Standards Association. *P802.1AS - Standard for Local and Metropolitan Area Networks - Timing and Synchronization for Time-Sensitive Applications*. Tech. rep. 2011 (cit. on p. 53).

[76] IEEE-Standards Association. *802.1Qav-2009 - IEEE Standard for Local and metropolitan area networks– Virtual Bridged Local Area Networks Amendment 12: Forwarding and Queuing Enhancements for Time-Sensitive Streams*. Tech. rep. 2009 (cit. on p. 53).

[77] IEEE-Standards Association. *802.1Qat-2010 - IEEE Standard for Local and metropolitan area networks–Virtual Bridged Local Area Networks Amendment 14: Stream Reservation Protocol (SRP)*. Tech. rep. 2010 (cit. on p. 53).

[78] B. Jesse and Vector Informatik GmbH. *Introduction of Audio/Video Bridging (AVB) over Ethernet in Vehicles – Embedded Software Architecture, Specifics and Use Cases*. 2015. URL: https://vector.com/portal/medien/cmc/events/Webinars/2015/Vector_Webinar_Audio_Video_Bridging_20150612_EN.pdf (visited on 6th Dec. 2016) (cit. on p. 53).

[79] TTTech Computertechnik. 'TTEthernet – A Powerful Network Solution for Multiple Purpose From Ethernet to TTEthernet'. In: *Deterministic Real-Time Ethernet Platform* (2016), pp. 1–14 (cit. on pp. 53, 54).

[80] M. Wolf, A. Weimerskirch and C. Paar. 'Security in Automotive Bus Systems'. In: *Workshop on Embedded IT-Security in Cars*. Bochum, Germany, Nov. 2004 (cit. on pp. 54, 55, 57–59).

[81] T. Hoppe and J. Dittmann. 'Sniffing/Replay Attacks on CAN Buses: A simulated attack on the electric window lift classified using an adapted CERT taxonomy'. In: *Proceedings of the 2nd Workshop on Embedded Systems Security (WESS)*. Salzburg, Austria, 2007 (cit. on pp. 54, 57).

[82] J. D. Howard and T. A. Longstaff. 'A Common Language for Computer Security Incidents'. In: Sandia Report: SAND98-8667 (1998). URL: http://www.cert.org/research/taxonomy_988667.pdf (cit. on p. 54).

[83] T. Hoppe, S. Kiltz and J. Dittmann. 'Security Threats to Automotive CAN Networks - Practical Examples and Selected Short-Term Countermeasures'. In: *Computer Safety, Reliability, and Security*. Ed. by M. D. Harrison and M.-A. Sujan. Lecture Notes in Computer Science 5219. Springer Berlin Heidelberg, Jan. 2008, pp. 235–248. ISBN: 978-3-540-87697-7, 978-3-540-87698-4. URL: http://link.springer.com/chapter/10.1007/978-3-540-87698-4_21 (visited on 19th Dec. 2016) (cit. on pp. 54, 57).

[84] D. K. Nilsson and U. E. Larson. 'Simulated Attacks on CAN Buses: Vehicle Virus'. In: *Proceedings of the 5th IASTED International Conference on Communication Systems and Networks*. AsiaCSN '08. Anaheim, CA, USA. Palma de Mallorca, Spain: ACTA Press, 2008, pp. 66–72. ISBN: 978-0-88986-758-1. URL: http://portal.acm.org/citation.cfm?id=1713277.1713292 (cit. on pp. 54, 57).

[85] T. Hoppe, S. Kiltz and J. Dittmann. 'Automotive IT-Security as a Challenge: Basic Attacks from the Black Box Perspective on the Example of Privacy Threats'. In: *Computer Safety, Reliability, and Security*. Ed. by B. Buth, G. Rabe and T. Seyfarth. Lecture Notes in Computer Science 5775. Springer Berlin Heidelberg, Jan. 2009, pp. 145–158. ISBN: 978-3-642-04467-0, 978-3-642-04468-7. URL: http://link.springer.com/chapter/10.1007/978-3-642-04468-7_13 (visited on 19th Dec. 2016) (cit. on pp. 55, 57).

[86] U. E. Larson and D. K. Nilsson. 'Securing Vehicles against Cyber Attacks'. In: *CSIIRW '08: Proceedings of the 4th annual workshop on Cyber security and information intelligence research*. CSIIRW '08. Proceedings of the 4th annual workshop on Cyber security and information intelligence research: developing strategies to meet the cyber security and information intelligence challenges ahead. New York, NY, USA: ACM, 2008, 30:1–30:3. ISBN: 978-1-60558-098-2. DOI: 10.1145/1413140.1413174. URL: http://doi.acm.org/10.1145/1413140.1413174 (cit. on pp. 55, 57, 60, 61).

[87] D. K. Nilsson and U. E. Larson. 'A Defense-in-Depth Approach to Securing the Wireless Vehicle Infrastructure'. In: *Journal of Networks* 4.7 (Sept. 2009), pp. 552–564. DOI: 10.4304/jnw.4.7.552-564. URL: http://academypublisher.com/jnw/vol04/no07/jnw0407552564.pdf (cit. on p. 55).

[88] M. L. Chávez, C. H. Rosete and F. R. Henríguez. 'Achieving Confidentiality Security Service for CAN'. In: *Proceedings of the 15th International Conference on Electronics, Communications and Computers, 2005. CONIELECOMP 2005*. Feb. 2005, pp. 166–170. DOI: 10.1109/CONIEL.2005.13 (cit. on pp. 56, 57).

[89] D. K. Nilsson, U. E. Larson, F. Picasso and E. Jonsson. 'A First Simulation of Attacks in the Automotive Network Communications Protocol FlexRay'. In: *Proceedings of the International Workshop on Computational Intelligence in Security for Information Systems (CISIS'08)*. Ed. by E. Corchado, R. Zunino, P. Gastaldo and Á. Herrero. Vol. 53. Advances in Intelligent and Soft Computing. 10.1007/978-3-540-88181-0_11. Springer Berlin / Heidelberg, 2009, pp. 84–91. URL: http://dx.doi.org/10.1007/978-3-540-88181-0_11 (cit. on p. 57).

[90] D. K. Nilsson, U. E. Larson and E. Jonsson. 'Efficient In-Vehicle Delayed Data Authentication Based on Compound Message Authentication Codes'. In: *Vehicular Technology Conference, 2008. VTC 2008-Fall. IEEE 68th*. 2008, pp. 1–5. DOI: 10.1109/VETECF.2008.259 (cit. on pp. 57, 60).

[91] A. Groll and C. Ruland. 'Secure and Authentic Communication on Existing In-Vehicle Networks'. In: *2009 IEEE Intelligent Vehicles Symposium*. 2009, pp. 1093–1097. DOI: 10.1109/IVS.2009.5164434 (cit. on pp. 57, 58).

[92] H. Oguma, A. Yoshioka, M. Nishikawa, R. Shigetomi, A. Otsuka and H. Imai. 'New Attestation-Based Security Architecture for In-Vehicle Communication'. In: *IEEE Global Telecommunications Conference, 2008. IEEE GLOBECOM 2008*. New Orleans, Louisiana, 2008, pp. 1–6. DOI: 10.1109/GLOCOM.2008.ECP.369 (cit. on pp. 57, 58).

[93] C. Szilagyi and P. Koopman. 'A Flexible Approach to Embedded Network Multicast Authentication'. In: *2nd Workshop on Embedded Systems Security (WESS)*. 2008 (cit. on pp. 57, 59).

[94] S. Schulze, M. Pukall, G. Saake, T. Hoppe and J. Dittmann. 'On the Need of Data Management in Automotive Systems'. In: *13. Fachtagung des GI-Fachbereichs "Datenbanken und Informationssysteme" (DBIS)*. Vol. 144. Gesellschaft für Informatik (GI). Münster, Germany, Mar. 2009 (cit. on pp. 57, 59).

[95] H. Schweppe, Y. Roudier, B. Weyl, L. Apvrille and D. Scheuermann. 'Car2X Communication: Securing the Last Meter - A Cost-Effective Approach for Ensuring Trust in Car2X Applications Using In-Vehicle Symmetric Cryptography'. In: *2011 IEEE Vehicular Technology Conference (VTC Fall)*. 2011, pp. 1–5. DOI: 10.1109/VETECF.2011.6093081 (cit. on pp. 57, 58).

[96] T. Hoppe, S. Kiltz and J. Dittmann. 'Adaptive Dynamic Reaction to Automotive IT Security Incidents Using Multimedia Car Environment'. In: *Proceedings of the 4th International Conference on Information Assurance and Security (ISIAS '08)*. Sept. 2008, pp. 295–298. DOI: 10.1109/IAS.2008.45 (cit. on pp. 57, 62, 63).

[97] T. Hoppe, S. Kiltz and J. Dittmann. 'Applying Intrusion Detection to Automotive IT — Early Insights and Remaining Challenges'. In: *Journal of Information Assurance and Security* 4.3 (2009), pp. 226–235. URL: http://www.mirlabs.org/jias/hoppe.pdf (cit. on pp. 57, 61, 63).

[98] M. Müter, A. Groll and F. C. Freiling. 'A Structured Approach to Anomaly Detection for In-Vehicle Networks'. In: *2010 Sixth International Conference on Information Assurance and Security (IAS)*. Atlanta, GA, Aug. 2010, pp. 92–98. DOI: 10.1109/ISIAS.2010.5604050 (cit. on pp. 57, 61).

[99] M. Müter and N. Asaj. 'Entropy-Based Anomaly Detection for In-Vehicle Networks'. In: *2011 IEEE Intelligent Vehicles Symposium (IV)*. Baden-Baden, Germany, June 2011, pp. 1110–1115. DOI: 10.1109/IVS.2011.5940552 (cit. on pp. 57, 62).

[100] R. Brooks, S. Sander, J. Deng and J. Taiber. 'Automobile Security Concerns'. In: *Vehicular Technology Magazine, IEEE* 4.2 (June 2009), pp. 52–64. ISSN: 1556-6072. DOI: 10.1109/MVT.2009.932539 (cit. on p. 57).

[101] G. Lee, H. Oguma, A. Yoshioka, R. Shigetomi, A. Otsuka and H. Imai. 'Formally Verifiable Features in Embedded Vehicular Security Systems'. In: *2009 IEEE Vehicular Networking Conference (VNC)*. Oct. 2009, pp. 1–7. DOI: 10.1109/VNC.2009.5416378 (cit. on p. 59).

[102] C. Szilagyi and P. Koopman. 'Flexible multicast authentication for time-triggered embedded control network applications'. In: *IEEE/IFIP International Conference on Dependable Systems Networks, 2009. DSN '09*. 2009, pp. 165–174. DOI: 10.1109/DSN.2009.5270342 (cit. on p. 60).

[103] P. Borazjani, C. Everett and D. McCoy. 'OCTANE: An Extensible Open Source Car Security Testbed'. In: *Proceedings of the Embedded Security in Cars Conference*. 2014 (cit. on p. 60).

[104] T. Hoppe, S. Kiltz and J. Dittmann. 'Security Threats to Automotive CAN Networks - Practical Examples and Selected Short-Term Countermeasures'. In: *Proceedings of the 27th International Conference on Computer Safety, Reliability, and Security (SAFECOMP '08)*. SAFECOMP '08. Newcastle upon Tyne, UK, Sept. 2008, pp. 235–248 (cit. on p. 61).

[105] T. Matsumoto, M. Hata, M. Tanabe, K. Yoshioka and K. Oishi. 'A Method of Preventing Unauthorized Data Transmission in Controller Area Network'. In: *Vehicular Technology Conference (VTC Spring), 2012 IEEE 75th*. May 2012, pp. 1–5. DOI: 10.1109/VETECS.2012.6240294 (cit. on p. 61).

[106] C. Ling and D. Feng. 'An Algorithm for Detection of Malicious Messages on CAN Buses'. In: *2012 National Conference on Information Technology and Computer Science. Atlantis Press*. 2012 (cit. on p. 62).

[107] K.-T. Cho and K. G. Shin. 'Fingerprinting Electronic Control Units for Vehicle Intrusion Detection'. In: *25th USENIX Security Symposium (USENIX Security 16)*. Austin, TX: USENIX Association, Aug. 2016, pp. 911–927. ISBN: 978-1-931971-32-4 (cit. on p. 63).

[108] V. Verendel, D. K. Nilsson, U. E. Larson and E. Jonsson. 'An Approach to using Honeypots in In-Vehicle Networks'. In: *Proceedings of the 68th IEEE Vehicular Technology Conference (VTC)*. Sept. 2008, pp. 1–5 (cit. on p. 63).

[109] M. Tim Jones and IBM Corporation. *Virtualization for embedded systems: The how and why of small-device hypervisors*. Apr. 2011. URL: https://www.ibm.com/developerworks/library/l-embedded-virtualization/ (visited on 12th Dec. 2016) (cit. on p. 66).

[110] K. Gandolfi, C. Mourtel and F. Olivier. 'Electromagnetic Analysis: Concrete Results'. English. In: *Cryptographic Hardware and Embedded Systems — CHES 2001*. Ed. by Ç. Koç, D. Naccache and C. Paar. Vol. 2162. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 2001, pp. 251–261. ISBN: 978-3-540-42521-2. DOI: 10.1007/3-540-44709-1_21. URL: http://dx.doi.org/10.1007/3-540-44709-1_21 (cit. on p. 66).

[111] Infineon Technologies AG. *AURIX™ Security Hardware*. 2016. URL: http://www.infineon.com/cms/en/product/microcontroller/32-bit-tricore-tm-microcontroller/aurix-tm-family/aurix-security-solutions/hardware/channel.html?channel=5546d4614815da8801487eb5b85d3291 (visited on 7th Dec. 2016) (cit. on pp. 68, 69).

[112] S. Aidanpää and E. Mk Nordmark. 'Flexible Updates of Embedded Systems Using Containers'. MA thesis. KTH Royal Institute of Technology, 2016, p. 112 (cit. on p. 69).

[113] AUTOSAR. *Specification of Module Secure Onboard Communication Release 4.2.2*. Tech. rep. 2015, p. 102 (cit. on pp. 70, 110).

[114] *ISO 26262 road vehicles functional safety part 1–10*. Standard. International Organization for Standardization, 2011 (cit. on pp. 71, 107).

[115] *SAE J3061 Cybersecurity Guide-book for Cyber-Physical Automotive Systems*. Standard. Vehicle Electrical System Security Committee, 2016 (cit. on pp. 71, 112).

[116] G. Macher, E. Armengaud, E. Brenner and C. Kreiner. 'A Review of Threat Analysis and Risk Assessment Methods in the Automotive Context'. In: *International Conference on Computer Safety, Reliability, and Security*. Springer. 2016, pp. 130–141 (cit. on p. 71).

[117] M. S. Lund, B. Solhaug and K. Stølen. *Model-driven risk analysis: the CORAS approach*. Springer Science & Business Media, 2010 (cit. on pp. 71, 74).

[118] T. UcedaVelez and M. M. Morana. *Risk Centric Threat Modeling: Process for Attack Simulation and Threat Analysis*. John Wiley & Sons, 2015 (cit. on pp. 72, 74).

[119] C. Alberts, A. Dorofee, J. Stevens and C. Woody. 'Introduction to the OCTAVE Approach'. In: *Pittsburgh, PA, Carnegie Mellon University* (2003) (cit. on pp. 72, 74).

[120] D. Mellado, E. Fernández-Medina and M. Piattini. 'Applying a security requirements engineering process'. In: *European Symposium on Research in Computer Security*. Springer. 2006, pp. 192–206 (cit. on pp. 72, 74).

[121] I. Flechais, M. A. Sasse and S. Hailes. 'Bringing security home: a process for developing secure and usable systems'. In: *Proceedings of the 2003 workshop on New security paradigms*. ACM. 2003, pp. 49–57 (cit. on pp. 72, 74).

[122] S. Mathew, M. Petropoulos, H. Q. Ngo and S. Upadhyaya. 'A data-centric approach to insider attack detection in database systems'. In: *International Workshop on Recent Advances in Intrusion Detection*. Springer. 2010, pp. 382–401 (cit. on p. 72).

[123] H. Mouratidis and P. Giorgini. 'Secure tropos: a security-oriented extension of the tropos methodology'. In: *International Journal of Software Engineering and Knowledge Engineering* 17.02 (2007), pp. 285–309 (cit. on pp. 72, 74).

[124] A. Van Lamsweerde. *Requirements engineering: from system goals to UML models to software specifications*. Wiley Publishing, 2009 (cit. on pp. 73, 74).

[125] M. Jackson. *Problem frames: analysing and structuring software development problems*. Addison-Wesley, 2001 (cit. on p. 73).

[126] D. Hatebur and M. Heisel. 'Problem frames and architectures for security problems'. In: *International Conference on Computer Safety, Reliability, and Security*. Springer. 2005, pp. 390–404 (cit. on pp. 73, 74).

[127] L. Lin, B. Nuseibeh, D. Ince and M. Jackson. 'Using abuse frames to bound the scope of security problems'. In: *Requirements Engineering Conference, 2004. Proceedings. 12th IEEE International*. IEEE. 2004, pp. 354–355 (cit. on pp. 73, 74).

[128] K. Stølen. 'CORAS A Framework for Risk Analysis of Security Critical Systems'. In: *supplement of the 2001 International Conference on Dependable Systems and Networks, pages D4-D11*. 2001 (cit. on p. 74).

[129] C. Woody, J. Coleman, M. Fancher, C. Myers and L. Young. *Applying octave: Practitioners report*. Tech. rep. DTIC Document, 2006 (cit. on p. 74).

[130] P. Kleberger. 'A Structured Approach to Securing the Connected Car'. In: (2012) (cit. on p. 74).

[131] P. Kleberger. *On Securing the Connected Car-Methods and Protocols for Secure Vehicle Diagnostics*. Chalmers University of Technology, 2015 (cit. on p. 74).

[132] D. Jani, D. Vanderveken and D. Perry. 'Deriving architecture specifications from KAOS specifications: a research case study'. In: *European Workshop on Software Architecture*. Springer. 2005, pp. 185–202 (cit. on p. 74).

[133] E. Casagrande, S. Woldeamlak, W. L. Woon, H. H. Zeineldin and D. Svetinovic. 'NLP-KAOS for systems goal elicitation: Smart metering system case study'. In: *IEEE Transactions on Software Engineering* 40.10 (2014), pp. 941–956 (cit. on p. 74).

[134] L.-C. Lin, B. Nuseibeh, D. Ince, M. Jackson and J. Moffett. 'Analysing security threats and vulnerabilities using abuse frames'. In: *ETAPS-04* (2003) (cit. on p. 74).

[135] *The STRIDE Threat Model*. https://msdn.microsoft.com/en-us/library/ee823878(v=cs.20).aspx. Accessed: 2016-11-25 (cit. on p. 74).

[136] P. Saitta, B. Larcom and M. Eddington. 'Trike v. 1 methodology document [draft]'. In: *URL: http://dymaxion. org/trike/Trike_v1_Methodology_Documentdraft. pdf* (2005) (cit. on p. 74).

[137] T. ETSI. '102 165-1 V4. 2.3 (2011-03)'. In: *Technical Specification Telecommunications and Internet converged Services and Protocols for Advanced Networking (TISPAN)* (2011) (cit. on p. 75).

[138] *HEAVENS: HEAling Vulnerabilities to ENhance Software Security and Safety*. http://www.vinnova.se/sv/Resultat/Projekt/Effekta/HEAVENS-HEAling-Vulnerabilities-to-ENhance-Software-Security-and-Safety/. Accessed: 2016-11-25 (cit. on pp. 75, 112).

[139] C. Schmittner, T. Gruber, P. Puschner and E. Schoitsch. 'Security application of failure mode and effect analysis (FMEA)'. In: *International Conference on Computer Safety, Reliability, and Security*. Springer. 2014, pp. 310–325 (cit. on p. 75).

[140] G. Macher, H. Sporer, R. Berlach, E. Armengaud and C. Kreiner. 'SAHARA: a security-aware hazard and risk analysis method'. In: *2015 Design, Automation & Test in Europe Conference & Exhibition (DATE)*. IEEE. 2015, pp. 621–624 (cit. on p. 75).

[141] A. V. Uzunov, E. B. Fernandez and K. Falkner. 'Engineering Security into Distributed Systems: A Survey of Methodologies.' In: *J. UCS* 18.20 (2012), pp. 2920–3006 (cit. on p. 77).

[142] P. H. Nguyen, M. Kramer, J. Klein and Y. Le Traon. 'An extensive systematic review on the Model-Driven Development of secure systems'. In: *Information and Software Technology* 68 (2015), pp. 62–81 (cit. on p. 77).

[143] A. van den Berghe, R. Scandariato, K. Yskout and W. Joosen. 'Design notations for secure software: a systematic literature review'. In: *Software & Systems Modeling* (2015), pp. 1–23 (cit. on p. 77).

[144] M. Alam, R. Breu and M. Hafner. 'Model-driven security engineering for trust management in SECTET'. In: *Journal of Software* 2.1 (2007), pp. 47–59 (cit. on p. 81).

[145] M. Memon, G. D. Menghwar, M. H. Depar, A. A. Jalbani and W. M. Mashwani. 'Security modeling for service-oriented systems using security pattern refinement approach'. In: *Software & Systems Modeling* 13.2 (2014), pp. 549–572 (cit. on p. 81).

[146] R. Breu, K. Burger, M. Hafner and G. Popp. 'Towards a Systematic Development of Secure Systems.' In: *Information Systems Security* 13.3 (2004), pp. 5–13 (cit. on p. 81).

[147] R. Breu, M. Hafner, F. Innerhofer-Oberperfler and F. Wozak. 'Model-driven security engineering of service oriented systems'. In: *International United Information Systems Conference*. Springer. 2008, pp. 59–71 (cit. on p. 81).

[148] M. Yague, E. Fernandez-Medina, M. Hafner, R. Breu, B. Agreiter and A. Nowak. 'SECTET: an extensible framework for the realization of secure inter-organizational workflows'. In: *Internet Research* 16.5 (2006), pp. 491–506 (cit. on p. 81).

[149] M. Hafner, M. Memon and R. Breu. 'SeAAS-A Reference Architecture for Security Services in SOA.' In: *J. UCS* 15.15 (2009), pp. 2916–2936 (cit. on p. 81).

[150] M. Hafner and R. Breu. *Security engineering for service-oriented architectures*. Springer Science & Business Media, 2008 (cit. on p. 81).

[151] C. Blanco, I. G.-R. de Guzmán, D. G. Rosado, E. Fernández-Medina, J. Trujillo et al. 'Applying QVT in order to implement secure data warehouses in SQL Server Analysis Services'. In: *Journal of Research and Practice in Information Technology* 41.2 (2009), p. 135 (cit. on p. 81).

[152] C. Blanco, E. Fernández-Medina and J. Trujillo. 'Modernizing secure OLAP applications with a model-driven approach'. In: *The Computer Journal* (2014), bxu070 (cit. on p. 81).

[153] C. Blanco, I. G. R. de Guzmán, E. Fernández-Medina and J. Trujillo. 'Showing the Benefits of Applying a Model Driven Architecture for Developing Secure OLAP Applications.' In: *J. UCS* 20.2 (2014), pp. 79–106 (cit. on p. 81).

[154] C. Blanco, R. Pérez-Castillo, A. Hernández, E. Fernández-Medina and J. Trujillo. 'Towards a modernization process for Secure Data Warehouses'. In: *International Conference on Data Warehousing and Knowledge Discovery*. Springer. 2009, pp. 24–35 (cit. on p. 81).

[155] E. Fernández-Medina and M. Piattini. 'Extending OCL for secure database development'. In: *International Conference on the Unified Modeling Language*. Springer. 2004, pp. 380–394 (cit. on p. 81).

[156] E. Fernández-Medina and M. Piattini. 'Designing secure databases'. In: *Information and Software Technology* 47.7 (2005), pp. 463–477 (cit. on p. 81).

[157] E. Soler, J. Trujillo, C. Blanco and E. Fernández-Medina. 'Designing Secure Data Warehouses by Using MDA and QVT.' In: *J. UCS* 15.8 (2009), pp. 1607–1641 (cit. on p. 81).

[158] J. Trujillo, E. Soler, E. Fernández-Medina and M. Piattini. 'A UML 2.0 profile to define security requirements for Data Warehouses'. In: *Computer Standards & Interfaces* 31.5 (2009), pp. 969–983 (cit. on p. 81).

[159] B. Vela, C. Blanco, E. Fernández-Medina and E. Marcos. 'Model driven development of secure XML data warehouses: a case study'. In: *Proceedings of the 2010 EDBT/ICDT Workshops*. ACM. 2010, p. 10 (cit. on p. 81).

[160] B. Vela, C. Blanco, E. Fernández-Medina and E. Marcos. 'A practical application of our MDD approach for modeling secure XML data warehouses'. In: *Decision Support Systems* 52.4 (2012), pp. 899–925 (cit. on p. 81).

[161] R. Villarroel, E. Fernández-Medina, M. Piattini and J. Trujillo. 'A UML 2.0/OCL extension for designing secure data warehouses'. In: *Journal of Research and Practice in Information Technology* 38.1 (2006), pp. 31–44 (cit. on p. 81).

[162] M. Borek, N. Moebius, K. Stenzel and W. Reif. 'Model-driven development of secure service applications'. In: *Software Engineering Workshop (SEW), 2012 35th Annual IEEE*. IEEE. 2012, pp. 62–71 (cit. on p. 81).

[163] N. Moebius, K. Stenzel and W. Reif. 'Modeling security-critical applications with UML in the SecureMDD approach'. In: *International Journal On Advances in Software* 1.1 (2008) (cit. on p. 81).

[164] N. Moebius, K. Stenzel, H. Grandy and W. Reif. 'Model-driven code generation for secure smart card applications'. In: *2009 Australian Software Engineering Conference*. IEEE. 2009, pp. 44–53 (cit. on p. 81).

[165] N. Moebius, K. Stenzel, M. Borek and W. Reif. 'Incremental development of large, secure smart card applications'. In: *Proceedings of the Workshop on Model-Driven Security*. ACM. 2012, p. 9 (cit. on p. 81).

[166] D. Basin, M. Clavel, J. Doser and M. Egea. 'A metamodel-based approach for analyzing security-design models'. In: *International Conference on Model Driven Engineering Languages and Systems*. Springer. 2007, pp. 420–435 (cit. on p. 81).

[167] D. Basin, M. Clavel, J. Doser and M. Egea. 'Automated analysis of security-design models'. In: *Information and Software Technology* 51.5 (2009), pp. 815–831 (cit. on p. 81).

[168] D. Basin, M. Clavel, M. Egea, M. A. G. de Dios, C. Dania, G. Ortiz and J. Valdazo. 'Model-driven development of security-aware GUIs for data-centric applications'. In: *Foundations of security analysis and design VI*. Springer, 2011, pp. 101–124 (cit. on p. 81).

[169] D. Basin, M. Clavel, M. Egea, M. A. G. de Dios and C. Dania. 'A model-driven methodology for developing secure data-management applications'. In: *IEEE Transactions on Software Engineering* 40.4 (2014), pp. 324–337 (cit. on p. 81).

[170] D. Basin, J. Doser and T. Lodderstedt. 'Model driven security: From UML models to access control infrastructures'. In: *ACM Transactions on Software Engineering and Methodology (TOSEM)* 15.1 (2006), pp. 39–91 (cit. on p. 81).

[171] C. Braga. 'A transformation contract to generate aspects from access control policies'. In: *Software & Systems Modeling* 10.3 (2011), pp. 395–409 (cit. on p. 81).

[172] M. Clavel, V. da Silva, C. Braga and M. Egea. 'Model-driven security in practice: An industrial experience'. In: *European Conference on Model Driven Architecture-Foundations and Applications*. Springer. 2008, pp. 326–337 (cit. on p. 81).

[173] M. A. G. de Dios, C. Dania, D. Basin and M. Clavel. 'Model-driven development of a secure eHealth application'. In: *Engineering Secure Future Internet Services and Systems*. Springer, 2014, pp. 97–118 (cit. on p. 81).

[174] J. Jürjens. 'UMLsec: Extending UML for secure systems development'. In: *International Conference on The Unified Modeling Language*. Springer. 2002, pp. 412–425 (cit. on p. 81).

[175] J. Jürjens. 'Using UMLsec and goal trees for secure systems development'. In: *Proceedings of the 2002 ACM symposium on Applied computing*. ACM. 2002, pp. 1026–1030 (cit. on p. 81).

[176] J. Jürjens and P. Shabalin. 'Tools for secure systems development with UML'. In: *International Journal on Software Tools for Technology Transfer* 9.5-6 (2007), pp. 527–544 (cit. on p. 81).

[177] J. Jürjens, L. Marchal, M. Ochoa and H. Schmidt. 'Incremental security verification for evolving UMLsec models'. In: *European Conference on Modelling Foundations and Applications*. Springer. 2011, pp. 52–68 (cit. on p. 81).

[178] J. Jürjens. 'Model-based security engineering with UML'. In: *Foundations of Security Analysis and Design III*. Springer, 2005, pp. 42–77 (cit. on p. 81).

[179] J. Abramov, O. Anson, M. Dahan, P. Shoval and A. Sturm. 'A methodology for integrating access control policies within database development'. In: *computers & security* 31.3 (2012), pp. 299–314 (cit. on p. 82).

[180] J. Abramov, O. Anson, A. Sturm and P. Shoval. 'Tool support for enforcing security policies on databases'. In: *Forum at the Conference on Advanced Information Systems Engineering (CAiSE)*. Springer. 2011, pp. 126–141 (cit. on p. 82).

[181] R. Bouaziz, S. Kallel and B. Coulette. 'An engineering process for security patterns application in component based models'. In: *Enabling Technologies: Infrastructure for Collaborative Enterprises (WETICE), 2013 IEEE 22nd International Workshop on*. IEEE. 2013, pp. 231–236 (cit. on p. 82).

[182] D.-K. Kim and P. Gokhale. 'A pattern-based technique for developing UML models of access control systems'. In: *30th Annual International Computer Software and Applications Conference (COMPSAC'06)*. Vol. 1. IEEE. 2006, pp. 317–324 (cit. on p. 82).

[183] D.-K. Kim, I. Ray, R. France and N. Li. 'Modeling role-based access control using parameterized UML models'. In: *International Conference on Fundamental Approaches to Software Engineering*. Springer. 2004, pp. 180–193 (cit. on p. 82).

[184] M. Schnjakin, M. Menzel and C. Meinel. 'A pattern-driven security advisor for service-oriented architectures'. In: *Proceedings of the 2009 ACM workshop on Secure web services*. ACM. 2009, pp. 13–20 (cit. on p. 82).

[185] S. Moral-García, S. Moral-Rubio, E. B. Fernández and E. Fernández-Medina. 'Enterprise security pattern: A model-driven architecture instance'. In: *Computer Standards & Interfaces* 36.4 (2014), pp. 748–758 (cit. on p. 82).

[186] D. G. Rosado, E. Fernandez-Medina, J. López and M. Piattini. 'PSecGCM: Process for the development of Secure Grid Computing based Systems with Mobile devices'. In: *Availability, Reliability and Security, 2008. ARES 08. Third International Conference on*. IEEE. 2008, pp. 136–143 (cit. on p. 82).

[187] D. G. Rosado, E. Fernández-Medina, J. López and M. Piattini. 'Systematic design of secure Mobile Grid systems'. In: *Journal of Network and Computer Applications* 34.4 (2011), pp. 1168–1183 (cit. on p. 82).

[188] D. Hatebur, M. Heisel and H. Schmidt. 'A security engineering process based on patterns'. In: *18th International Workshop on Database and Expert Systems Applications (DEXA 2007)*. IEEE. 2007, pp. 734–738 (cit. on p. 82).

[189] D. Hatebur, M. Heisel and H. Schmidt. 'A pattern system for security requirements engineering'. In: *Availability, Reliability and Security, 2007. ARES 2007. The Second International Conference on*. IEEE. 2007, pp. 356–365 (cit. on p. 82).

[190] D. Hatebur, M. Heisel and H. Schmidt. 'Analysis and component-based realization of security requirements'. In: *Availability, Reliability and Security, 2008. ARES 08. Third International Conference on*. IEEE. 2008, pp. 195–203 (cit. on p. 82).

[191] H. Schmidt. *A pattern-and component-based method to develop secure software*. Deutscher Wissenschaftsverlag, 2010 (cit. on p. 82).

[192] H. Schmidt. 'Threat-and Risk-Analysis During Early Security Requirements Engineering.' In: *ARES*. 2010, pp. 188–195 (cit. on p. 82).

[193] H. Schmidt, D. Hatebur and M. Heisel. 'A pattern-based method to develop secure software'. In: *Software Engineering for Secure Systems: Industrial and Research Perspectives: Industrial and Research Perspectives* (2010), p. 32 (cit. on p. 82).

[194] A. Maña, F. Sanchez-Cid, D. Serrano and A. Muñoz. 'Towards Secure Ambient Intelligence Scenarios.' In: *SEKE*. 2006, pp. 386–391 (cit. on p. 82).

[195] G. Spanoudakis and S. Kokolakis. *Security and Dependability for Ambient Intelligence*. Vol. 45. Springer Science & Business Media, 2009 (cit. on p. 82).

[196] C. Gutiérrez, E. Fernández-Medina and M. Piattini. 'Towards a process for web services security'. In: *Journal of Research and Practice in Information Technology* 38.1 (2006), pp. 57–68 (cit. on p. 82).

[197] C. Gutiérrez, D. G. Rosado and E. Fernández-Medina. 'The practical application of a process for eliciting and designing security in web service systems'. In: *Information and Software Technology* 51.12 (2009), pp. 1712–1738 (cit. on p. 82).

[198] M. Almorsy and J. Grundy. 'Secdsvl: a domain-specific visual language to support enterprise security modelling'. In: *2014 23rd Australian Software Engineering Conference*. IEEE. 2014, pp. 152–161 (cit. on p. 83).

[199] M. Almorsy, J. Grundy and A. S. Ibrahim. 'Adaptable, model-driven security engineering for SaaS cloud-based applications'. In: *Automated software engineering* 21.2 (2014), pp. 187–224 (cit. on p. 83).

[200] Y. Elrakaiby, M. Amrani and Y. Le Traon. 'Security@ runtime: A flexible mde approach to enforce fine-grained security policies'. In: *International Symposium on Engineering Secure Software and Systems*. Springer. 2014, pp. 19–34 (cit. on p. 83).

[201] B. Morin, T. Mouelhi, F. Fleurey, Y. Le Traon, O. Barais and J.-M. Jézéquel. 'Security-driven model-based dynamic adaptation'. In: *Proceedings of the IEEE/ACM international conference on Automated software engineering*. ACM. 2010, pp. 205–214 (cit. on p. 83).

[202] S. Gilmore, L. Gönczy, N. Koch, P. Mayer, M. Tribastone and D. Varró. 'Non-functional properties in the model-driven development of service-oriented systems'. In: *Software & Systems Modeling* 10.3 (2011), pp. 287–311 (cit. on p. 83).

[203] M. Menzel, R. Warschofsky and C. Meinel. 'A pattern-driven generation of security policies for service-oriented architectures'. In: *Web Services (ICWS), 2010 IEEE International Conference on*. IEEE. 2010, pp. 243–250 (cit. on p. 83).

[204] M. Menzel and C. Meinel. 'Securesoa modelling security requirements for service-oriented architectures'. In: *Services Computing (SCC), 2010 IEEE International Conference on*. IEEE. 2010, pp. 146–153 (cit. on p. 83).

[205] C. Wolter, M. Menzel, A. Schaad, P. Miseldine and C. Meinel. 'Model-driven business process security requirement specification'. In: *Journal of Systems Architecture* 55.4 (2009), pp. 211–223 (cit. on p. 83).

[206] Y. Nakamura, M. Tatsubori, T. Imamura and K. Ono. 'Model-driven security based on a web services security architecture'. In: *2005 IEEE International Conference on Services Computing (SCC'05) Vol-1*. Vol. 1. IEEE. 2005, pp. 7–15 (cit. on p. 83).

[207] G. Georg, I. Ray, K. Anastasakis, B. Bordbar, M. Toahchoodee and S. H. Houmb. 'An aspect-oriented methodology for designing secure applications'. In: *Information and Software Technology* 51.5 (2009), pp. 846–864 (cit. on p. 83).

[208] D. Mouheb, C. Talhi, M. Nouh, V. Lima, M. Debbabi, L. Wang and M. Pourzandi. 'Aspect-oriented modeling for representing and integrating security concerns in UML'. In: *Software Engineering Research, Management and Applications 2010*. Springer, 2010, pp. 197–213 (cit. on p. 83).

[209] I. Ray, R. France, N. Li and G. Georg. 'An aspect-based approach to modeling access control concerns'. In: *Information and Software Technology* 46.9 (2004), pp. 575–587 (cit. on p. 83).

[210] ó. Sánchez, F. Molina, J. García-Molina and A. Toval. 'ModelSec: a generative architecture for model-driven security'. In: *J. Univ. Comput. Sci* 15.15 (2009), pp. 2957–2980 (cit. on p. 83).

[211] G.-J. Ahn and H. Hu. 'Towards realizing a formal RBAC model in real systems'. In: *Proceedings of the 12th ACM symposium on Access control models and technologies*. ACM. 2007, pp. 215–224 (cit. on p. 84).

[212] S. Kim, D.-K. Kim, L. Lu, S. Kim and S. Park. 'A feature-based approach for modeling role-based access control systems'. In: *Journal of Systems and Software* 84.12 (2011), pp. 2035–2052 (cit. on p. 84).

[213] K. Sohr, G.-J. Ahn, M. Gogolla and L. Migge. 'Specification and validation of authorisation constraints using UML and OCL'. In: *European Symposium on Research in Computer Security*. Springer. 2005, pp. 64–79 (cit. on p. 84).

[214] M. Koch, L. V. Mancini and F. Parisi-Presicce. 'A graph-based formalism for RBAC'. In: *ACM Transactions on Information and System Security (TISSEC)* 5.3 (2002), pp. 332–365 (cit. on p. 84).

[215] M. Koch and F. Parisi-Presicce. 'UML specification of access control policies and their formal verification'. In: *Software & Systems Modeling* 5.4 (2006), pp. 429–447 (cit. on p. 84).

[216] J. A. Pavlich-Mariscal, S. A. Demurjian and L. D. Michel. 'A framework of composable access control features: Preserving separation of access control concerns from models to code'. In: *Computers & Security* 29.3 (2010), pp. 350–379 (cit. on p. 84).

[217] L. Yu, R. France, I. Ray and S. Ghosh. 'A Rigorous Approach to Uncovering Security Policy Violations in UML Designs'. In: *Engineering of Complex Computer Systems, 2009 14th IEEE International Conference on*. IEEE. 2009, pp. 126–135 (cit. on p. 84).

[218] C. C. Burt, B. R. Bryant, R. R. Raje, A. Olson and M. Auguston. 'Model driven security: unification of authorization models for fine-grain access control'. In: *Enterprise Distributed Object Computing Conference, 2003. Proceedings. Seventh IEEE International*. IEEE. 2003, pp. 159–171 (cit. on p. 84).

[219] T. Fink, M. Koch and K. Pauls. 'An MDA approach to access control specifications using MOF and UML profiles'. In: *Electronic Notes in Theoretical Computer Science* 142 (2006), pp. 161–179 (cit. on p. 84).

[220] T. Mouelhi, F. Fleurey, B. Baudry and Y. Le Traon. 'A model-based framework for security policy specification, deployment and testing'. In: *International Conference on Model Driven Engineering Languages and Systems*. Springer. 2008, pp. 537–552 (cit. on p. 84).

[221] A. Kaddani, A. Baina and L. Echabbi. 'Towards a Model Driven Security for critical infrastructures using OrBAC'. In: *Multimedia Computing and Systems (ICMCS), 2014 International Conference on*. IEEE. 2014, pp. 1235–1240 (cit. on p. 84).

[222] A. Bertolino, M. Busch, S. Daoudagh, F. Lonetti and E. Marchetti. 'A toolchain for designing and testing access control policies'. In: *Engineering Secure Future Internet Services and Systems*. Springer, 2014, pp. 266–286 (cit. on p. 84).

[223] M. Eby, J. Werner, G. Karsai and A. Ledeczi. 'Integrating security modeling into embedded system design'. In: *14th Annual IEEE International Conference and Workshops on the Engineering of Computer-Based Systems (ECBS'07)*. IEEE. 2007, pp. 221–228 (cit. on p. 84).

[224] P. Diaz, I. Aedo, D. Sanz and A. Malizia. 'A model-driven approach for the visual specification of Role-Based Access Control policies in web systems'. In: *2008 IEEE Symposium on Visual Languages and Human-Centric Computing*. IEEE. 2008, pp. 203–210 (cit. on p. 84).

[225] G. Graham and B. Cohen. *Microsoft Security Development Lifecycle Adoption*. Microsoft, Edison Group. Sept. 2013 (cit. on pp. 85, 89).

[226] *ISO/IEC 27034: Information technology — Security techniques — Application security — Part 1: Overview and concepts*. Standard ISO/IEC 27034-1:2011. International Organization for Standardization (cit. on p. 85).

[227] Cisco Systems, Inc. *Cisco Secure Development Lifecycle (CSDL)*. URL: http://www.cisco.com/web/about/security/cspo/csdl/index.html (visited on 22nd Nov. 2013) (cit. on p. 87).

[228] Foundstone (McAfee). *Secure Software Development Lifecycle (SSDLC)*. 2008. URL: http://www.mcafee.com/us/resources/data-sheets/foundstone/ds-secure-software-dev-life-cycle.pdf (visited on 22nd Nov. 2013) (cit. on p. 88).

[229] Microsoft. *Secure Development Lifecycle (SDL)*. URL: http://www.microsoft.com/security/sdl/resources/publications.aspx (visited on 22nd Nov. 2013) (cit. on pp. 89, 90).

[230] OWASP (Open Web Application Security Project). *Open Software Assurance Maturity Model (SAMM), Version 1.0*. Mar. 2009. URL: http://www.opensamm.org/downloads/SAMM-1.0.pdf (visited on 2nd Dec. 2016) (cit. on p. 90).

[231] G. McGraw, S. Migues and J. West. *BSIMM-V (Building Security In Maturity Model, Version 5.0)*. Oct. 2013. URL: http://www.bsimm.comm/download/dl.php (visited on 22nd Nov. 2013) (cit. on pp. 90, 91).

[232] OWASP (Open Web Application Security Project). *Comprehensive, Lightweight Application Security Process (CLASP)*. http://www.owasp.org. Accessed: 2016-10-28. (Visited on 28th Oct. 2016) (cit. on pp. 91, 98).

[233] P. Hellström. *Master's Thesis: Tools for Static Code Analysis: A Survey*. 2009. DOI: LIU-IDA/LITH-EX-A--09/003--SE (cit. on pp. 92–94).

[234] Microsoft. *SDL Threat Modeling*. URL: http://www.microsoft.com/security/sdl/adopt/threatmodeling.aspx (visited on 25th Nov. 2013) (cit. on p. 94).

[235] CERT. *CERT Basic Fuzzing Framework*. URL: http://www.cert.org/vuls/discovery/bff.html (visited on 25th Nov. 2013) (cit. on p. 94).

[236] Microsoft. *SDL MiniFuzz File Fuzzer*. URL: http://www.microsoft.com/en-us/download/details.aspx?id=21769 (visited on 25th Nov. 2013) (cit. on p. 94).

[237] Microsoft. *SDL Regex Fuzzer*. URL: http://www.microsoft.com/en-us/download/details.aspx?id=20095 (visited on 25th Nov. 2013) (cit. on p. 95).

[238] B. De Win, R. Scandariato, K. Buyens, J. Grégoire and W. Joosen. 'On the secure software development process: CLASP, SDL and Touchpoints compared'. In: *Information and software technology* 51.7 (2009), pp. 1152–1171 (cit. on pp. 95, 96).

[239] M. Howard and S. Lipner. *The security development lifecycle*. Vol. 8. Microsoft Press Redmond, 2006 (cit. on p. 96).

[240] G. McGraw. *Software security: building security in*. Vol. 1. Addison-Wesley Professional, 2006 (cit. on p. 98).

[241] K. R. van Wyk and G. McGraw. 'Bridging the gap between software development and information security'. In: *IEEE Security & Privacy* 3.5 (2005), pp. 75–79 (cit. on p. 99).

[242] CAR 2 CAR Communication Consortium. *C2C-CC Manifesto*. v1.1. Aug. 2007. URL: http://www.car-to-car.org/ (visited on 19th Dec. 2016) (cit. on p. 101).

[243] G. Karagiannis, O. Altintas, E. Ekici, G. Heijenk, B. Jarupan, K. Lin and T. Weil. 'Vehicular Networking: A Survey and Tutorial on Requirements, Architectures, Challenges, Standards and Solutions'. In: *IEEE Communications Surveys & Tutorials* 13.4 (2011), pp. 584–616. DOI: 10.1109/SURV.2011.061411.00019 (cit. on p. 101).

[244] T. L. Willke, P. Tientrakool and N. F. Maxemchuk. 'A Survey of Inter-Vehicle Communication Protocols and Their Applications'. In: *IEEE Communications Surveys & Tutorials* 11.2 (2009), pp. 3–20. DOI: 10.1109/SURV.2009.090202 (cit. on p. 101).

[245] M. L. Sichitiu and M. Kihl. 'Inter-Vehicle Communication Systems: A Survey'. In: *IEEE Communications Surveys & Tutorials* 10.2 (2008), pp. 88–105. DOI: 10.1109/COMST.2008.4564481 (cit. on p. 101).

[246] 'Trust assurance levels of cybercars in v2x communication'. In: *Proceedings of the 2013 ACM workshop on Security, privacy & dependability for cyber vehicles* (2013), pp. 49–60. ISSN: 15437221. DOI: 10.1145/2517968.2517974. URL: http://dl.acm.org/citation.cfm?id=2517974 (cit. on p. 102).

[247] K. Dar, M. Bakhouya, J. Gaber, M. Wack and P. Lorenz. 'Wireless Communication Technologies for ITS Applications'. In: *IEEE Communications Magazine* 48.5 (2010), pp. 156–162. DOI: 10.1109/MCOM.2010.5458377 (cit. on p. 104).

[248] A. Avivzienis, J.-C. Laprie, B. Randell and C. Landwehr. 'Basic Concepts and Taxonomy of Dependable and Secure Computing'. In: *IEEE Transactions on Dependable and Secure Computing* 1.1 (2004), pp. 11–33. DOI: 10.1109/TDSC.2004.2 (cit. on p. 105).

[249] IEC. *International Electrotechnical Commission (IEC)*. URL: http://www.iec.ch (visited on 19th Dec. 2016) (cit. on p. 108).

[250] ETSI. *Telecommunications and Internet converged Services and Protocols for Advanced Networking (TISPAN); Methods and protocols; Part 1: Method and proforma for Threat, Risk, Vulnerability Analysis*. Technical Specification TS 102 165-1, v4.2.3. 650 Route des Lucioles, F-06921 Sophia Antipolis Cedex, France: ETSI, Mar. 2011 (cit. on p. 109).

[251] EUCAR. *European Council for Automotive R&D (EUCAR)*. URL: http://www.eucar.be (visited on 19th Dec. 2016) (cit. on p. 109).

[252] NHTSA. *National Highway Traffic Safety Administration (NHTSA)*. URL: http://www.nhtsa.gov (visited on 19th Dec. 2016) (cit. on p. 110).

[253] AUTOSAR Consortium. *AUTomotive Open System ARchitecture*. URL: https://www.autosar.org/ (visited on 14th Nov. 2016) (cit. on p. 110).

[254] CARONTE Project. *Creating an Agenda for Research On Transportation sEcurity*. URL: http://www.caronte-project.eu/ (visited on 14th Nov. 2016) (cit. on p. 111).

[255] N. Nowdehi and T. Olovsson. 'Experiences from implementing the ETSI ITS SecuredMessage service'. In: *IEEE Intelligent Vehicles Symposium, Proceedings* Iv (2014), pp. 1055–1060. DOI: 10.1109/IVS.2014.6856587 (cit. on p. 112).

[256] SESAMO. *Security and Safety Modeling (SESAMO)*. URL: http://www.sesamo-project.eu (visited on 19th Dec. 2016) (cit. on p. 113).

[257] CORDIS - Community Research and Development Information Service. *SAFURE - SAFety and secURity by design for interconnected mixed-critical cyber-physical systems*. URL: http://cordis.europa.eu/project/rcn/194149_en.html (visited on 8th Dec. 2016) (cit. on p. 113).

[258] CORDIS - Community Research and Development Information Service. *SafeCOP - Safe Cooperating Cyber-Physical Systems using Wireless Communication*. URL: http://cordis.europa.eu/project/rcn/203404_en.html (visited on 8th Dec. 2016) (cit. on p. 114).

[259] CORDIS - Community Research and Development Information Service. *SCOUT - Safe and COnnected aUtomation in road Transport*. URL: http://cordis.europa.eu/project/rcn/204978_en.html (visited on 8th Dec. 2016) (cit. on p. 114).

[260] CORDIS - Community Research and Development Information Service. *SHARCS - Secure Hardware-Software Architectures for Robust Computing Systems*. URL: http://cordis.europa.eu/project/rcn/194217_en.html (visited on 8th Dec. 2016) (cit. on p. 114).

[261] CORDIS - Community Research and Development Information Service. *CYRail - Cybersecurity in the RAILway sector*. URL: http://cordis.europa.eu/project/rcn/206014_en.html (visited on 8th Dec. 2016) (cit. on p. 114).

[262] CORDIS - Community Research and Development Information Service. *IMMORTAL - Integrated Modelling, Fault Management, Verification and Reliable Design Environment for Cyber-Physical Systems*. URL: http://cordis.europa.eu/project/rcn/194260_en.html (visited on 8th Dec. 2016) (cit. on p. 115).

