

# Protecting Data Communication for the Next Generation Electric Vehicles

Zur Erlangung des akademischen Grades Doktor-Ingenieur (Dr.-Ing.)  
Genehmigte Dissertation von Daniel Zelle aus Minden  
Tag der Einreichung: 13. Mai 2024, Tag der Prüfung: 10. Juli 2024

1. Gutachten: Prof. Dr. Michael Waidner
  2. Gutachten: Prof. Dr. Frank Kargl
  3. Gutachten: Prof. Dr. Christoph Krauß
- Darmstadt, Technische Universität Darmstadt



TECHNISCHE  
UNIVERSITÄT  
DARMSTADT

Computer Science  
Department



**Fraunhofer**  
SIT

Fraunhofer Institute for  
Secure Information  
Technology

Cyberphysical Systems  
Security

Protecting Data Communication for the Next Generation Electric Vehicles

Accepted doctoral thesis by Daniel Zelle

Date of submission: 13. Mai 2024

Date of thesis defense: 10. Juli 2024

Darmstadt, Technische Universität Darmstadt

Bitte zitieren Sie dieses Dokument als:

URN: urn:nbn:de:tuda-tuprints-280374

URL: <https://tuprints.ulb.tu-darmstadt.de/28037>

Jahr der Veröffentlichung auf TUprints: 2024

Dieses Dokument wird bereitgestellt von tuprints,

E-Publishing-Service der TU Darmstadt

<https://tuprints.ulb.tu-darmstadt.de>

[tuprints@ulb.tu-darmstadt.de](mailto:tuprints@ulb.tu-darmstadt.de)

Die Veröffentlichung steht unter folgender Creative Commons Lizenz:

Namensnennung 4.0 International

<https://creativecommons.org/licenses/by/4.0/>

This work is licensed under a Creative Commons License:

Attribution 4.0 International

<https://creativecommons.org/licenses/by/4.0/>

---

## Erklärungen laut Promotionsordnung

### § 8 Abs. 1 lit. c PromO

Ich versichere hiermit, dass die elektronische Version meiner Dissertation mit der schriftlichen Version übereinstimmt.

### § 8 Abs. 1 lit. d PromO

Ich versichere hiermit, dass zu einem vorherigen Zeitpunkt noch keine Promotion versucht wurde. In diesem Fall sind nähere Angaben über Zeitpunkt, Hochschule, Dissertationsthema und Ergebnis dieses Versuchs mitzuteilen.

### § 9 Abs. 1 PromO

Ich versichere hiermit, dass die vorliegende Dissertation selbstständig und nur unter Verwendung der angegebenen Quellen verfasst wurde.

### § 9 Abs. 2 PromO

Die Arbeit hat bisher noch nicht zu Prüfungszwecken gedient.

Darmstadt, 13. Mai 2024

---

D. Zelle



---

# Zusammenfassung

---

Diese Dissertation untersucht das vielschichtige Gebiet der Cybersicherheit von modernen Elektrofahrzeugen (EV) mit einem Schwerpunkt auf Fahrzeugkommunikationsprotokollen. Unsere Forschung beleuchtet detailliert IT-Sicherheitsrisiken, wie die Manipulation des Fahrverhaltens, Datenschutzverletzungen während des Ladevorgangs und Bedrohungen für die Stabilität des Stromnetzes durch das Laden von EVs.


In dieser Arbeit identifizieren wir durch eine Bedrohungs- und Risikoanalyse (TARA) identifizieren die kritischen Angriffspunkte in der Fahrzeugkommunikation. Wir führen dabei einen innovativen Ansatz zur automatisierten Bewertung der Angriffsfläche ein, um den TARA-Prozess zu beschleunigen und Fehler zu reduzieren. Diese Bewertung ergibt hohe Risiken sowohl bei der internen Fahrzeug- als auch bei der Ladekommunikation.

Zur Absicherung der internen Kommunikation via CAN-Bus wird das BusCount-Protokoll zum Schutz von Fahrzeugnetzwerken vorgestellt. Es bietet gegenüber bestehenden Lösungen deutliche Vorteile beim Schutz vor Replay- und Delay-Angriffen.

Weiterhin validieren wir die Absicherung des modernen Automotive Ethernet mittels TLS anhand verschiedener Kommunikationsszenarien. Wir messen die Leistungsauswirkungen verschiedener Chiffren auf typischer Automotive-Hardware und vergleichen sie mit den Anforderungen der Automobilindustrie. Für service-orientierten Kommunikation über Automotive Ethernet weisen wir Sicherheitslücken des verbreiteten SOME/IP Protokolls nach und entwickeln zwei mögliche Protokollerweiterungen zur Absicherung.

Zur Sicherung der EV-Ladeinfrastruktur gehen wir auf potenzielle Manipulation des Stromnetzes und Datenschutzprobleme während des Ladevorgangs ein. Weiterhin zeigen wir, wie personenbezogenen Daten beim Ladevorgang reduziert werden können und schlagen eine Erweiterung für das bestehende Plug & Charge Protokolle vor, die ein DAA-Schema nutzt, um einen anonymisierten Ladevorgang zu ermöglichen.

Zusammenfassend trägt diese Arbeit zur Weiterentwicklung der Cybersicherheit von EVs bei, indem kritische Aspekte durch eine Risikobewertung identifiziert und Lösungen



---

zur Absicherung der internen CAN- und Automotive Ethernet-Kommunikation vorgeschlagen werden. Darüber hinaus werden Sicherheits- und Datenschutzprobleme der EV-Ladeinfrastruktur adressiert. Diese Erkenntnisse bilden eine Grundlage für die IT-Sicherheit in Elektrofahrzeuge in einer sich schnell entwickelnden Automobilindustrie.

---

# Abstract

---

This dissertation focuses on the cybersecurity of modern electric vehicles (EVs), specifically their communication protocols. The research highlights IT security risks, such as the manipulation of driving behavior, data breaches during the charging process, and threats to the stability of the power grid from EV charging.


A threat and risk analysis (TARA) is conducted to identify the critical points of attack in vehicle communication. We present an innovative approach to automating the assessment of the attack surface, which will accelerate the TARA process and reduce errors. The assessment reveals high risks in internal vehicle communication as well as in charging communication.

To secure internal communication via the CAN bus, we propose the BusCount protocol as a secure communication solution for protecting automotive networks. This approach provides clear advantages over existing solutions in protecting against replay and delay attacks.

Furthermore, we investigate securing modern automotive Ethernet using TLS and validate various communication scenarios. We assess the potential performance impacts of different ciphers on typical automotive hardware and compare them to the industry's requirements. Security gaps were identified in the widespread SOME/IP protocol in service-oriented communication via automotive Ethernet. We developed two possible protocol extensions to secure SOME/IP.

This work also addresses potential manipulation of the power grid and data protection issues during the charging process to ensure the security of the EV charging infrastructure. We show how to reduce the transmission of personal data during the charging process and propose an extension for the existing Plug & Charge protocols. Our proposed solution utilizes a Direct Anonymous Attestation (DAA) scheme for anonymous charging.

In summary, this work contributes to advancing EV cybersecurity by identifying critical aspects through a risk assessment and proposing ways to secure internal Controller Area



---

Network (CAN) and automotive Ethernet communication. Additionally, this work addresses security and privacy issues related to the EV charging infrastructure. These findings and solutions provide a solid foundation for creating a more secure environment for EVs in the rapidly evolving automotive industry.



---

# Acknowledgments

---

First and foremost, I would like to express my sincere gratitude to my supervisors, Prof. Dr. Michael Waidner, Prof. Dr. Frank Kargl, and Prof. Dr. Christoph Krauß, for their invaluable guidance and support throughout my PhD journey. Their insightful feedback, constant encouragement, and unwavering belief in my work have been instrumental in completing this thesis. I am also grateful to the members of my thesis committee, Prof. Faust, Ph. D., and Prof. Dr. Fischlin, for their valuable feedback and suggestions, which significantly improved the quality of this work.

I would like to thank my present and former colleagues and students at Fraunhofer SIT who contributed to an open and supportive research environment by working together on inventive and inspiring projects and openly exchanging thoughts, ideas, and arguments. My special thanks go to my co-authors, Dr. Roland Rieke and Dr. Sigrid Gürgens, who introduced me to formal security analysis and helped me improve my scientific work. Furthermore, I would like to thank Christian Plappert, Dustin Kern, Timm Lauser, and Norman Lahr, with whom I worked on multiple problems in vehicle security. A special thanks to Markus Springer, Maria Zhdanova, Jonathan Stancke, Jens Gutmann, and Marius Wilch, with whom I explored the new field of e-mobility and built awesome demonstrators.

I would like to thank all the collaborating partners in the different projects I could participate in. Their expertise and contributions have been crucial to the success of this research. Financial support from the Federal Ministry for Economic Affairs and Climate Action through the project unIT-e2 (01MV21UN12) as well as the Federal Ministry of Education and Research and Hessen State Ministry for Higher Education, Research and the Arts through the National Research Center for Applied Cybersecurity ATHENE.

I am deeply indebted to my family for their unwavering love and support throughout this challenging journey. Their encouragement and belief in me have kept me motivated during difficult times. I want to give a special thanks to my wife, Katharine, who has always been there for me, offering a listening ear and words of encouragement. I am truly grateful for her persistent support.



---

# Contents

---

<b>1. Introduction</b>	<b>1</b>
1.1. Motivation . . . . .	1
1.2. Problem Statement . . . . .	3
1.3. Goal . . . . .	3
1.4. Contribution . . . . .	4
1.5. Structure . . . . .	8
<b>2. Background</b>	<b>11</b>
2.1. Controller Area Network (CAN) . . . . .	11
2.2. Automotive Ethernet . . . . .	12
2.2.1. Scalable service-Oriented MiddlewarE over IP (SOME/IP) . . . . .	13
2.2.2. Diagnostics over Internet Protocol (DoIP) . . . . .	14
2.3. Charging Infrastructure for Electric Vehicles (EVs) . . . . .	15
2.3.1. Plug-and-Charge Architecture . . . . .	16
2.3.2. ISO 15118 . . . . .	16
2.3.3. Open Charge Point Protocol (OCPP) . . . . .	20
2.3.4. Open Intercharge Protocol (OICP) . . . . .	20
<b>3. System Model</b>	<b>23</b>
3.1. Vehicular Network . . . . .	23
3.2. Communication Scenarios Electric Vehicles . . . . .	26
3.2.1. In-vehicle Communication . . . . .	26
3.2.2. External Communication . . . . .	28
3.2.3. Diagnostics . . . . .	28
3.3. Attacker Model . . . . .	29
3.3.1. Local Attacker . . . . .	29
3.3.2. Advanced Remote Attacker . . . . .	29
3.3.3. Internal Attacker . . . . .	30

---

<b>4. Risk Analysis of Next Generation Electric Vehicles</b>	<b>31</b>
4.1. The ISO/SAE 21434 Threat Analysis and Risk Assessment (TARA) Process	32
4.2. Attack Surface Assessment	32
4.2.1. Asset Definition in Modern Electric Vehicles	33
4.2.2. Attack Feasibility Rating	35
4.2.3. Discussion on Different Path Calculation Methods	43
4.3. Automated Attack Path Generation	44
4.3.1. Example	45
4.3.2. Directed Weighted Graph Generation	45
4.3.3. Example	45
4.3.4. Attack Path Extraction	47
4.3.5. Example	48
4.3.6. Threat Path Extraction	48
4.3.7. Most Feasible Path Calculation	49
4.4. Related Work on Threat and Risk Analysis	50
4.5. Application to Reference Architecture	51
4.5.1. Transferring Reference Architecture to Graph	52
4.5.2. Identifying Entry Points	52
4.5.3. Generating Attack Paths	52
4.5.4. Identifying Attacker Goals	54
4.5.5. Identifying Feasible Attack Paths	56
4.5.6. Feasible Attack Paths against Vehicle Safety	57
4.5.7. Feasible Attack Paths against Charging Communication	58
4.6. Summary of Risk Assessment for Modern Electric Vehicle Reference Architecture	59
<b>5. Controller Area Network (CAN) Bus Security</b>	<b>61</b>
5.1. Setting	62
5.1.1. Protection Goals	63
5.1.2. State of the Art CAN Bus Security	64
5.2. The Generic Counter Concept	66
5.2.1. Formal Security Analysis of Generic Counter Concept	68
5.3. BusCount: A Low Layer Bus Counter Solution	70
5.3.1. Synchronization	70
5.3.2. Formal Verification of BusCount	72
5.4. Evaluation	74
5.4.1. Security Evaluation	74
5.4.2. Practical Aspects	75

---

5.5. Summary on CAN Bus Security . . . . .	79
<b>6. Automotive Ethernet Security</b>	<b>81</b>
6.1. Setting of Automotive Ethernet Network . . . . .	82
6.2. Introducing Transport Layer Security (TLS) to In-Vehicle Communication .	83
6.2.1. Requirements Analysis . . . . .	84
6.2.2. Discussion on the Applicability of TLS to Secure In-Vehicle Networks	87
6.2.3. Performance Analysis . . . . .	92
6.2.4. Discussion of Measured Results . . . . .	100
6.2.5. Related Work on Securing Automotive Ethernet . . . . .	102
6.3. Secure Service Oriented Automotive Communication . . . . .	103
6.3.1. Network Security Assumptions . . . . .	105
6.3.2. Related Work on Securing Service-oriented Communication . . . .	105
6.3.3. Formal Security Analysis Approach . . . . .	106
6.3.4. SOME/IP Analysis and Attacks . . . . .	108
6.3.5. Attack Evaluation . . . . .	114
6.3.6. Security Extensions . . . . .	118
6.4. Summary on Automotive Ethernet Security . . . . .	128
<b>7. Improved Security and Privacy for Charging Systems</b>	<b>131</b>
7.1. Basics on Direct Anonymous Attestation (DAA) . . . . .	132
7.2. Related Work of the Security of EV Charging . . . . .	133
7.3. Security and Privacy Analysis of EV Charging Protocols . . . . .	135
7.3.1. Attack Vectors . . . . .	136
7.3.2. Security Analysis of EV Charging . . . . .	136
7.3.3. Privacy Analysis of EV Charging . . . . .	139
7.4. V2G Attack Strategies . . . . .	147
7.4.1. Attacks on EVs and Charge Stations (CSs) . . . . .	147
7.4.2. Malicious Backend Attack . . . . .	149
7.4.3. Attacks on Charging Communications . . . . .	150
7.5. Attack Examples against Charging Protocols . . . . .	152
7.5.1. OCPP Charging Schedule Attack . . . . .	152
7.5.2. ISO 15118 Man-in-the-Middle (MitM) Attack . . . . .	153
7.6. Practical Security Analysis of V2G Communication . . . . .	155
7.6.1. Evaluation of OCPP Communication . . . . .	155
7.6.2. Evaluation of ISO 15118 Communication . . . . .	156
7.6.3. Relevance of the Evaluation Results . . . . .	158

---

7.7. Limitations and Mitigation . . . . .	158
7.7.1. Practicality of the Attack . . . . .	158
7.7.2. Mitigation Strategies . . . . .	160
7.8. Privacy-Enhancing Charging Solution . . . . .	161
7.8.1. Requirements Analysis for a Privacy-Enhancing Charging Solution .	161
7.8.2. Proposed Privacy Extension for EV Charging . . . . .	164
7.8.3. Security Discussion on the Anonymous Charging Protocol . . . . .	168
7.9. Comparison with State of the Art Anonymous Charging . . . . .	171
7.10. Summary on the Security of Charging Infrastructure . . . . .	172
<b>8. Conclusion and Future Work</b>	<b>173</b>
<b>A. Appendix</b>	<b>203</b>
A.1. Formal Security of CAN . . . . .	204
A.1.1. Formal Security Analysis of Generic Counter Concept . . . . .	204
A.1.2. Formal Verification of BusCount . . . . .	207
A.2. Formal Security of SOME/IP . . . . .	213
A.2.1. Tamarin model of SOME/IP . . . . .	213
A.2.2. Tamarin model of SESO-RC . . . . .	226
A.2.3. Tamarin model of SESO-AS . . . . .	232

---

# 1. Introduction

---

As vehicles evolve into complex cyber-physical systems with advanced features and communication protocols, protecting them from emerging cyber threats is crucial. This extensive inquiry explores this pressing need at the dynamic intersection of automotive engineering and cutting-edge information technology. This thesis analyzes cyber security vulnerabilities linked to high-risk communication in electric vehicles and proposes security measures.

## 1.1. Motivation

Information Technology (IT) is one of the main drivers for innovation in modern vehicles and is of paramount importance towards autonomous vehicles. Next to autonomous driving vehicles, the introduction of connected cars and EVs are further trends. Due to the shift to electric engines, the engine's complexity is reduced, and battery management and charging are introduced. Thus, manufacturers introduced new communication protocols. At the same time, new, more capable in-vehicle network communication technologies have been introduced to vehicle architecture to transmit high-resolution sensor signals necessary for autonomous or assisted driving. Moreover, advanced infotainment systems and a connection unit, the so-called Telematic Control Unit (TCU), have been introduced, enabling many connected driving features. Examples are remote updates, live traffic information, remote parking, emergency calls, or online entertainment.

Traditionally, a vehicle network, the so-called E/E architecture, is separated into domains, each with a Controller Area Network (CAN) bus network connecting Electronic Control Units (ECUs) for a specific purpose. For example, the power or drive domain connects ECUs that control the engine and brakes. The comfort domain contains side mirrors, windows, electronic seat adjustments, and climate control controllers. An infotainment domain with a radio, navigation system, and instrument cluster may also use a Media

---

Oriented Systems Transport (MOST) bus for message communication. All domains are interconnected with domain controllers via CAN.

The general idea of this separation in in-vehicle network architectures remains mostly the same in the next generation of vehicles; however, the technologies used can change. With these innovative technologies, the domains are consolidated into a smaller number of domains. One emerging trend in modern in-vehicle communication is utilizing network technologies with enhanced bandwidth, such as Automotive Ethernet (100 Mbit/s) [36], to replace the traditional CAN bus network, which interconnects the domains. Furthermore, the traditional CAN bus network technology (1 Mbit/s) evolves towards a higher bandwidth with its predecessors Controller Area Network Flexible Data-Rate (CAN FD) (5 Mbit/s) [62] and Controller Area Network Extra Long (CAN XL) (10 Mbit/s) [5].

Introducing these changed network technologies in the architecture involves how data communication is treated in vehicles. Currently, ECUs send a cyclic signal to indicate the status of a system. For example, each car transfers the message `rear light off` every 50 milliseconds if the vehicle is not in reverse gear. This signal-based communication is now partially replaced by service-based communication, allowing request-response communication known from IP-based networks. The automotive industry developed the middleware protocol Scalable service-Oriented MiddlewarE over IP (SOME/IP) [6, 7] for Automotive Ethernet to communicate between ECUs. Besides SOME/IP, the automotive industry has also introduced Audio Video Bus (AVB) for time-synchronous video and audio transmission. Moreover, Diagnostics over Internet Protocol (DoIP) [92] implements diagnostic functions and update mechanisms for vehicle repair shops. An overview of new protocols is given in [124].

Vehicles also communicate with external entities beyond the local communication in the in-vehicle network. Modern cars have multiple connections to external devices such as the vehicle key, smartphones via Bluetooth or Wi-Fi, the manufacturer's backend via cellular communication (like LTE, GSM, or 5G), or other vehicles via Vehicular Ad Hoc Network (VANet). Moreover, mechanics connect tester devices via On-Board Diagnostics (OBD) for vehicle diagnostics [31]. Electric vehicles introduce a new connection and communication to the charge point for recharging batteries directly to the vehicle network.

This communication with ISO 15118 allows the exchange of charging parameters as well as the accounting of energy. The charging and billing of electric vehicles are currently realized with different (primarily incompatible) systems, e.g., using multiple Radio Frequency Identification (RFID) cards or smartphone apps. Almost all charging systems require some registration with a service provider, and every charging session involves not only an EV and a CS but also multiple backend systems of energy providers, power network



---

---

operators, vehicle manufacturers, regulators, and Mobility Operator (MO). For the unification of charging and billing, so-called Plug-and-Charge (PnC) protocols, such as ISO 15118 [89, 90, 91], Open Charge Point Protocol (OCPP) [140, 141], and Open Interchange Protocol (OICP) [142], were developed. Next to billing, these protocols also allow vehicle smart charging to manage power consumption and maintain a stable power grid.

## 1.2. Problem Statement

All these protocols and technologies increase the attack surface for hackers, especially since information security is not a key focus when these are developed. Many prominent attacks on vehicle networks and the surrounding infrastructure have already been shown in the past. The academic demonstration of attacks on in-vehicle networks started with [106] by controlling driving functionalities by injecting CAN messages. The message injection can also occur via the diagnostics port OBD [59]. Next to these direct attacks on network communication, several attacks show the vulnerability of vehicle ECUs. Examples are the Jeep hack [128] and attacks on the Tesla Model S [137], Volkswagen [101], BMW [27], Mercedes-Benz [200] and KIA [37]. Dalheimer has exposed attacks on the charging infrastructure [40], demonstrating the ability to control different CSs and the unencrypted communication between CSs and the corresponding backend system.

## 1.3. Goal

My overall goal for this thesis is to increase the security of vehicle communication by showing the weaknesses of current vehicle networks with several newly introduced technologies and introducing my security concepts into the vehicle architecture. To achieve this, I have developed the following:

- Systematically analyzes upcoming communication technologies of next-generation EVs and their security goals.
- Development of attacks against these communication protocols.
- Introduction and evaluation of countermeasures for the presented vulnerabilities.

---

In the first step, we systematically analyze upcoming and present communication technologies of next-generation EVs. Based on this analysis, we determine a cyber attacker's attack feasibility and risks. For this purpose, we design a generic reference architecture of modern EVs to evaluate the impacts of attacks. Next to the necessary reference architecture, we establish an attacker model and attack vectors to derive the security properties of automotive communication. A key focus of our analysis is security guarantees of protocols regarding attacks that alter or delay messages for in-vehicle communication as well as deriving of personal data for external vehicle communication. In-vehicle communication includes low-level CAN bus, typically used for communication between ECUs in one domain, and high-level Automotive Ethernet communication, primarily for inter-domain communication. Both are safety-critical communication technologies. On the one hand, communication to and in a charging infrastructure via OCPP and ISO 15118 contains a personal identifier, accounting information, and positioning and driving behavior of a person. On the other hand, this communication allows controlling the charging process of an EV, which has potential impacts on the power grid with catastrophic consequences for grid stability. All this information is communicated via multiple entities. Furthermore, this communication influences the charging schedule, which impacts the power consumed. This communication can also affect the power grid. Well-discussed automotive protocols like VANet and classic Ethernet protocols like Dynamic Host Configuration Protocol (DHCP) or Precision Time Protocol (PTP) are not part of this thesis.

## 1.4. Contribution

This thesis significantly enhances vehicle security in multiple ways. The following delineates this work's contributions to the advancement of the state of the art, along with their respective impacts. Additionally, each chapter of the thesis clearly outlines my contributions to the papers published, distinguishing them from those of my co-authors.

**Generic Domain-based Vehicle Architecture** Initially, we introduce a generic domain-based vehicle reference architecture designed to analyze vehicle communication security while considering the current vehicle architectures of different Original Equipment Manufacturers (OEMs). We also create attacker models derived from the real-world attacks of the past years. This architecture was published in [207] and forms the basis of all further contributions. The paper not only provides a realistic architecture but also gathers components to implement our test bench, which we use in this thesis to simulate attacks

---

or implement security solutions. The test bench is the first academic platform using automotive Ethernet. It integrates autonomous driving as well as safety functionalities specially designed for evaluating attacks and security technologies on a model car with real physical impacts. So far, most similar platforms focus on simulations to train autonomous driving; thus, these do not mimic realistic network architectures with distributed sensors and actors.

- [207] Daniel Zelle et al. “SEPAD - Security Evaluation Platform for Autonomous Driving”. In: *28th Euromicro International Conference on Parallel, Distributed and Network-Based Processing, PDP 2020, Västerås, Sweden, March 11-13, 2020*. IEEE, 2020, pp. 413–420. DOI: 10.1109/PDP50117.2020.00070. URL: <https://doi.org/10.1109/PDP50117.2020.00070>

**Automated Attack Feasibility Rating** Next, we introduce an automated attack feasibility rating mechanism for the newly established ISO 21434. This mechanism closes the gap in ISO 21434 between the definition of the assets and potential-based rating, where the ISO standard only gives high-level advice, by introducing a novel method to automatically calculate potential-based attack feasibility based on an attack model and basic attack steps. The procedure was published in [208] and is the first publication considering an OEM’s requirements. The risk analysis was initially developed for and in cooperation with a German OEM that uses these results internally for further risk treatment decisions. Multiple research projects in the industry have been acquired due to this topic, and our training content has been updated to ISO 21434. The automatism of this approach reduces the danger of human mistakes in the rating process. It thus generates attack vectors more reliably based on the network model and basic attack steps. Furthermore, it reduces the workload when creating a threat and risk analysis of vehicles or components.

- [208] Daniel Zelle et al. “ThreatSurf: A method for automated Threat Surface assessment in automotive cybersecurity engineering”. In: *Microprocessors and Microsystems* 90 (2022), p. 104461. ISSN: 0141-9331. DOI: <https://doi.org/10.1016/j.micpro.2022.104461>. URL: <https://www.sciencedirect.com/science/article/pii/S0141933122000321>

**Security Improvement of Controller Area Network** Furthermore, we discovered a new security vulnerability in representative security protocols for CAN using formal security verification and propose a solution to mitigate this issue. With the introduction of formal

---

verification [67, 202], we expose that most security protocols for CAN, including Automotive Open System Architecture (AUTOSAR)’s Secure Onboard Communication (SecOC), send messages that can be monitored and altered by an attacker. This allows the attacker to send this message at a later point in time, and potential receivers accept it. We present our provably secure protocol extension for the data link layer of CAN that is compatible with regular CAN controllers as well as its successors, CAN FD and CAN XL. This problem and its solutions have been integrated into industry projects with multiple tier-one suppliers in the automotive industry.

- [202] Daniel Zelle, Sigrid Gürgens, and Anjia Yang. “BusCount: A Provable Replay Protection Solution for Automotive CAN Networks”. In: *Security and Communication Networks 2021* (Jan. 2021). ISSN: 1939-0114. DOI: 10.1155/2021/9951777. URL: <https://doi.org/10.1155/2021/9951777>
- [67] Sigrid Gürgens and Daniel Zelle. “A Hardware Based Solution for Freshness of Secure Onboard Communication in Vehicles”. In: *Computer Security - ESORICS 2018 International Workshops, CyberICPS 2018 and SECPRE 2018, Barcelona, Spain, September 6-7, 2018, Revised Selected Papers*. Ed. by Sokratis K. Katsikas et al. Vol. 11387. Lecture Notes in Computer Science. Springer, 2018, pp. 53–68. ISBN: 978-3-030-12785-5. DOI: 10.1007/978-3-030-12786-2\_4. URL: [https://doi.org/10.1007/978-3-030-12786-2\\_4](https://doi.org/10.1007/978-3-030-12786-2_4)

**Security Improvement of Automotive Ethernet** Moreover, we conduct a systematic analysis of automotive Ethernet, demonstrating the feasibility of TLS in a vehicle network for the first time. The papers [206] have been developed in collaboration with a German OEM, resulting in the adoption of TLS for in-vehicle communication and discussions on securing in-vehicle networks with further OEMs and suppliers. We also present the first systematic review of security attacks and countermeasures for Automotive Ethernet in ACM Computing Surveys [189] covering 172 papers. Additionally, we identify new vulnerabilities through the first formal security verification of SOME/IP in combination with low-layer security protocols widely used in automotive networks. These vulnerabilities could be verified within the reference implementation and a standard development environment in the automotive industry. Due to the dynamic service-oriented communication, the protocol allows attackers to attack the service discovery and gain a man-in-the-middle position in all communication scenarios of SOME/IP, giving the attacker complete control over the transmitted messages. To overcome these vulnerabilities, we propose and implement two security extensions for SOME/IP; one is designed for improved performance, and the other offers perfect forward secrecy. For both, we formally prove the security properties

---

of both protocols. The paper [203] presented this and received the Best Paper award at ARES '21. Furthermore, this research resulted in evaluations of OEMs and suppliers to introduce further security technologies for their SOME/IP implementations.

- [203] Daniel Zelle et al. “Analyzing and Securing SOME/IP Automotive Services with Formal and Practical Methods”. In: *ARES 2021: The 16th International Conference on Availability, Reliability and Security, Vienna, Austria, August 17-20, 2021*. Ed. by Delphine Reinhardt and Tilo Müller. ACM, 2021, 8:1–8:20. DOI: 10.1145/3465481.3465748. URL: <https://doi.org/10.1145/3465481.3465748> (Best Paper Award)
- [206] Daniel Zelle et al. “On Using TLS to Secure In-Vehicle Networks”. In: *Proceedings of the 12th International Conference on Availability, Reliability and Security, Reggio Calabria, Italy, August 29 - September 01, 2017*. ACM, 2017, 67:1–67:10. ISBN: 978-1-4503-5257-4. DOI: 10.1145/3098954.3105824. URL: <https://doi.org/10.1145/3098954.3105824>
- [43] Marco De Vincenzi et al. “A Systematic Review on Security Attacks and Countermeasures in Automotive Ethernet”. In: *ACM Comput. Surv.* 56.6 (Jan. 2024). ISSN: 0360-0300. DOI: 10.1145/3637059. URL: <https://doi.org/10.1145/3637059>

**Security and Privacy Analysis of Electric Vehicle (EV) Charging** Finally, we identify security and privacy issues in current charging protocols, demonstrating these vulnerabilities through practical evaluations and developing an evaluation tool for CSs. Findings in specific products have been reported to the charge station manufacturers. In [204], we first examined the protocols of electric charging systematically to find all personal data transmissions in the complete communication chain from the vehicle to the MO. Based on the analysis of current charging protocols, with several privacy flaws, we developed the first extension to ISO 15118 to overcome these problems using DAA that maintains the ISO architecture and does not introduce additional actors but allows an exact accounting of charging processes in the complex relationship between multiple Charge Station Operators (CSOs), and MOs and their costumers. This extension uses DAA to authenticate the user towards a CS anonymously and encrypts power consumption between CS and MO, preventing a clearing house from learning the actual communication. Moreover, we presented [212] at the Annual Computer Security Applications Conference (ACSAC) 2022, describing possible attack scenarios on power grids based on security flaws shown in an experimental evaluation of different attack techniques specially fitted for EV charging.

---

The findings and results lead to multiple invitations due to our expertise regarding the security and privacy of the current charging infrastructure. For example, with Electrify America, Nationale Leitstelle Ladeinfrastruktur, CharIN Cybersecurity Workshop, and a German OEM.

- [204] Daniel Zelle et al. “Anonymous Charging and Billing of Electric Vehicles”. In: *Proceedings of the 13th International Conference on Availability, Reliability and Security, ARES 2018, Hamburg, Germany, August 27-30, 2018*. Ed. by Sebastian Doerr et al. ACM, 2018, 22:1–22:10. ISBN: 978-1-4503-6448-5. DOI: 10.1145/3230833.3230850. URL: <https://doi.org/10.1145/3230833.3230850>
- [212] Maria Zhdanova et al. “Local Power Grids at Risk - An Experimental and Simulation-based Analysis of Attacks on Vehicle-To-Grid Communication”. In: *Annual Computer Security Applications Conference, ACSAC 2022, Austin, TX, USA, December 5-9, 2022*. ACM, 2022, pp. 42–55. DOI: 10.1145/3564625.3568136. URL: <https://doi.org/10.1145/3564625.3568136>

## 1.5. Structure

The thesis is structured into seven chapters. In the next chapter (Chapter 2), we introduce the fundamentals of comprehending the communication technologies used in modern vehicular networks and the electric vehicle charging infrastructure with its diverse communication media and protocols. In the following Chapter 3, we describe the developed reference architecture of modern electric vehicles, which details the connections between different entities in the cars and the corresponding charging infrastructure. We also describe the typical communication scenarios of modern vehicles. Based on existing attacks in this chapter, we then establish three different attackers relevant to the IT security of cars. Based on the reference architecture, we discuss the risks associated with communication vehicle data traffic in Chapter 4. For this purpose, a self-developed scheme based on ISO 21434 [208] is used to evaluate and assess attack paths in the vehicle network. Based on these results, a selection of problems is addressed in the following chapters, starting with protecting low-level can bus communication in Chapter 5 against attackers. In the Chapter 6, we explain possible attacks and security concepts for Automotive Ethernet. First, a general approach is presented to secure vehicular communication with standard TLS, including a possible certificate architecture and performance evaluations of TLS on an automotive processor. Next, the service-oriented protocol SOME/IP is analyzed, and security issues are discovered that cannot be solved by applying TLS. Two provable secure

---

solutions are presented to overcome the security issue of the multicast service discovery process. In the following chapter (Chapter 7), we discuss the security and privacy issues of the charging communication. First, the security implications of EV smart charging on the power grid are elaborated. Then, the approach is presented, ensuring end-to-end privacy of charging processes. In the end, a conclusion is given of the presented results.





---

## 2. Background

---

This chapter gives an overview of standard communication technologies in a vehicle network. We start with CAN bus and Automotive Ethernet communication. We will also examine EV charging, the backend communication necessary for the charging and billing process, and the scheduling of charging processes and control mechanisms to increase the power grid's stability.

### 2.1. Controller Area Network (CAN)

Controller Area Network (CAN) bus is a field bus commonly used in automotive networks [86]. In this bus network, all connected devices are able to send and receive messages with up to 1 Mbit/s transfer rate.

Each regular CAN message comprises the parts: "Start of frame" bit, message identifier, control field, data field, checksum, confirmation field, and "end of frame". Such a message is shown in Figure 2.1.

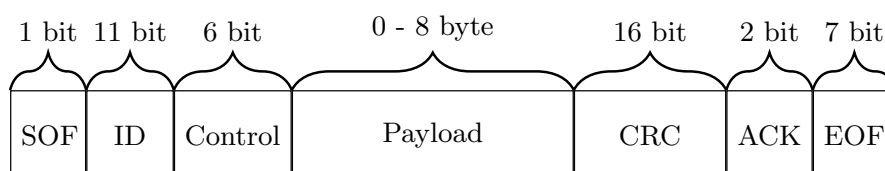


Figure 2.1.: A single CAN frame

After the first bit indicating the message starts, an 11-bit identifier is transferred with every message, which stands for its priority and allows it to handle collisions using Carrier Sense Multiple Access/Collision Resolution (CSMA/CR). CSMA/CR prevents collisions with a simple procedure: all entities in a network send messages simultaneously, indicated

---

by the start of a frame. Due to the physical properties of the CAN bus, a dominant bit (0) overwrites a recessive bit (1). Each entity monitors the bus, and in case its message ID is overwritten, it stops sending. This means the lowest ID is sent without a collision.

The following control bits contain the data length, as well as a reserved bit and a bit indicating that the message ID has been extended to an additional 18 bits.

After the payload of up to 8 bytes, a Cyclic Redundancy Check (CRC) over the message is transmitted and checked by every ECU in the network. In case the CRC check has failed, or another problem occurs, each entity in the network aware of the problem sends an error frame that invalidates the message for all receivers. This action increases the error counter by 8 for the sender and 1 for every receiver. An entity reaching an error count of 128 is disabled and can no longer interact with the network. A successful message transmission decreases the error counter, but only if the counter did not reach 128 before.

Since the transmission of 8 bytes is no longer sufficient, Controller Area Network Flexible Data-Rate (CAN FD) [86] and Controller Area Network Extra Long (CAN XL) [5] have been developed. Both differ mainly in the bit rate used to transfer the payload. CAN FD transmits up to 64 bytes, while CAN XL is capable of a 2048-byte message payload in a single CAN frame.

## 2.2. Automotive Ethernet

Automotive Ethernet is based on the BroadR-Reach-Ethernet-Physical-Layer [81], which was later specified by the OPEN Alliance SIG<sup>1</sup> and IEEE in various standards. Automotive Ethernet allows the connection of ECUs via unshielded single twisted pair cables, reducing the wires' weight and cost. It provides communication with up to 1000 Mbit/s in an IEEE 802.3 switch network, and thus, classic Ethernet communication via IP and UDP or TCP is possible. Automotive Ethernet replaced parts of classical vehicle network technology, especially high bandwidth systems like MOST and FlexRay. A typical Automotive Ethernet scenario is described in [56]. It contains a low number of control units with Ethernet ports (between 10 and 20) and one or two Automotive Ethernet gateways or switches connecting these ECUs. On the application layer, some automotive-specific protocols have been introduced. Examples are SOME/IP or DoIP. These two application layer protocols, SOME/IP and DoIP, for Automotive Ethernet are discussed in the following.

---

<sup>1</sup><https://opensig.org/about/specifications/>

---

## 2.2.1. Scalable service-Oriented MiddlewarE over IP (SOME/IP)

With the introduction of Automotive Ethernet, signal-oriented communication has been partially complemented with service-oriented communication. The Scalable service-Oriented MiddlewarE over IP (SOME/IP) protocol, specified in AUTOSAR [6], is designed to provide service-oriented communication and acts as a middleware between hardware and software components.

SOME/IP utilizes the TCP/IP or UDP/IP protocol to transport data (cf. Figure 2.2) and provides a serialization of control signals. Applications that use the SOME/IP interface do not require IP addresses or ports. However, only a service identifier is needed to communicate, making it a more flexible and bandwidth-efficient communication protocol. Furthermore, its design elements provide the necessary features to facilitate communication efficiently.

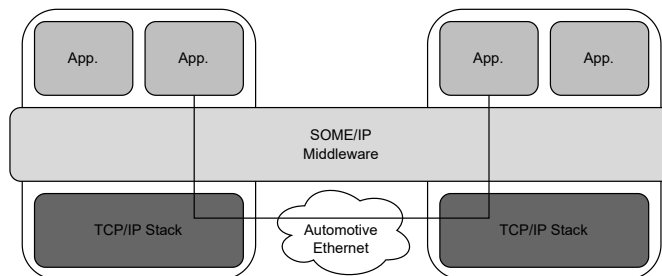


Figure 2.2.: SOME/IP Architecture

To allow communication between entities based on service IDs, SOME/IP makes use of Service Discovery (SD). Scalable service-Oriented MiddlewarE over IP (SOME/IP-SD) implements the detection of service instances and introduces a publish/subscribe process [7]. A service instance offers by sending a multicast message. Since multiple instances can provide the same service in a network, each instance is uniquely identified by an Instance ID. This is called offer message `OfferService(Service ID, Instance ID)`, containing the endpoint options (IP address, IP Version, protocol, and port), service and instance ID, and options for configurations load balancing. By sending a `StopOffer(Service ID, Instance ID)` message, a server indicates that the service is no longer offered. A client sends a `FindService(Service ID, Instance ID)` message to search for a service actively. If the client only looks for a specific instance, the ID can be set to `0xFFFF`. A server that provides the requested service responds with an offer service message.

SOME/IP offers different communication patterns depending on the type of service. Clients can request a service and get a response from the server but also send a message to trigger

a service provider function without any response message; this communication pattern is called fire and forget. A client can also subscribe to a service that either notifies the client regularly or when an event occurs. To subscribe to a service, also called an event group, a client sends a `SubscribeEventgroup(Service ID, Instance ID)` message to the server offering the service. The server confirms the subscription with `SubscribeEventgroupACK(Service ID, Instance ID)`. A client transmits a `StopSubscribeEventgroup(Service ID, Instance ID)` message to unsubscribe from an event group. Figure 2.3 displays the regular information flow of this subscription process.

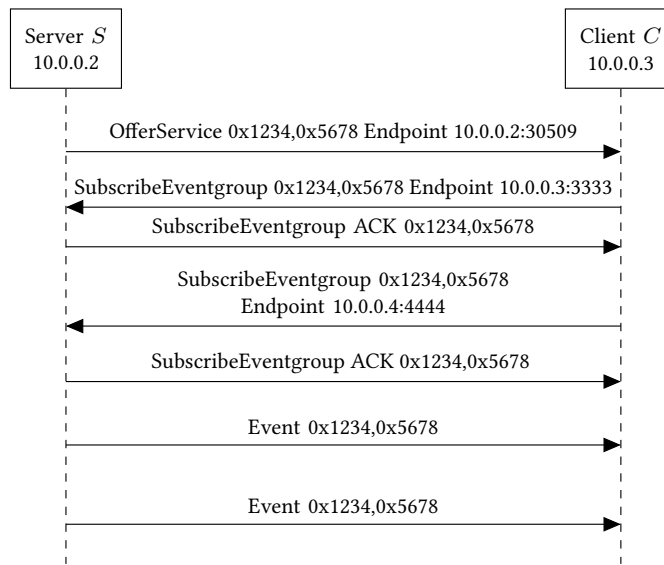


Figure 2.3.: Typical SOME/IP publish subscribed protocol flow

Both protocol specifications, SOME/IP [6] and SOME/IP-SD [7], do not provide any security mechanisms. A typical SOME/IP network offers about ten services per ECU, according to [56]

### 2.2.2. Diagnostics over Internet Protocol (DoIP)

A further Automotive Ethernet protocol, Diagnostics over Internet Protocol (DoIP), is specially designed for the particular use of vehicle diagnostics and troubleshooting in a repair shop. DoIP [92] is a transportation layer protocol designed to enable communication

---

between automotive control units and testers. In the typical case, a testing tool is connected to the vehicle via an OBD port and allows a mechanic to read error codes and vehicle data. Furthermore, modern cars have an integrated tester that allows remote diagnostics and updates by the vehicle manufacturer. An additional use case of DoIP enables the connection of multiple testers with one or multiple vehicles via a wireless network in a repair shop. This makes the work of mechanics more convenient and eliminates the need for physical connections to the vehicle.

The DoIP protocol is divided into two phases: The discovery phase and the actual diagnostic communication to transfer, e.g., sensor values or trouble codes. In the initialization phase, a DoIP gateway broadcasts a `VehicleAnnouncementmessage` announcing its presence to connected testers. The tester can also actively search for the gateway by sending a `VehicleIdentificationRequest`, which the gateway answers with a `VehicleIdentificationResponse`. In the following routing phase, the tester requests the gateway to open a socket to receive further commands. Once the communication is set up, the tester performs its diagnostics requests. Internally, the gateway can use DoIP or proprietary protocols to get information from other control units in the vehicle.

### **2.3. Charging Infrastructure for Electric Vehicles (EVs)**

A unique communication of an EV is the charging communication. This communication allows the vehicle to communicate with Charge Station (CS) and exchange parameters for the charging process. Furthermore, a schedule for the charging can be set up. It also authenticates the power transfer and accounts for the charging process. Charging communication is not only limited to communication between the CS and the vehicle but also includes communication between the CS and backend systems and between backend systems. Lots of protocols for EV charging have been proposed [149, 136, 72], but only a small set is deployed in actual products. The primary standards are ISO 15118 [89] for the communication between vehicle and CS, Open Charge Point Protocol (OCPP) [140, 141] by the Open Charge Alliance that enables remote control capabilities to manage multiple chargers located at different sites and the accounting of charging processes, and Open Intercharge Protocol (OICP) [142] that implements an interface between acCSO and the MO, which sells energy to a customer.

---

### 2.3.1. Plug-and-Charge Architecture

Based on the information given on the most common or upcoming charging protocols and their role definition in Section 2.3, as well as the information on charging infrastructure given in [205] and [73], a reference Plug-and-Charge (PnC) architecture is given.

In this architecture, a customer owns an EV and has a contract with an MO that allows the customer to charge his vehicle at public CSs. For this purpose, the vehicle communicates with the CS and authenticates using Plug-and-Charge. The CS forwards the authorization request to the operator CSO using OCPP. In case the CSO cannot answer the request directly, it will contact the Contract Clearing House (CCH) using OICP, which sends this request to the corresponding MO. The response by the MO is then sent via the same channels, and the vehicle can start charging. This charging process might be reduced or interrupted by the Distribution System Operator (DSO). The DSO can send control signals or tariff information to the CSO or a local Energy Management System (EMS), and both can contact the CS using OCPP to re-schedule the charging process. This new charging schedule is transmitted via Power-Line Communication (PLC) and ISO 15118 to the vehicle. In the end, when the charging process is finished, the CS transmits the final meter value to the car and the CSO. The CSO transmits the values to the CCH, and from there, the values are sent to the MO, who sends a bill to the customer and pays the CSO for the consumed energy. Figure 2.4 gives an overview of the reference architecture.

The assumed system architecture is based on the definitions of ISO 15118 [89, 90, 91], OCPP 1.6 [140], and OICP 2.3 [142].

### 2.3.2. ISO 15118

ISO 15118 [89, 90, 91] is an international standard defining a Vehicle to Grid (V2G) communication interface supported by a large number of OEMs and CS manufacturers [50]. The goal of ISO 15118 is to enable interoperability between different CSs and vehicles by plugging in the power cable without further interaction.

To understand the complex charging infrastructure, it is essential first to know the different actors and roles defined in [89]. Afterwards, the communication protocol is described.

The first party is the customer, who charges an EV at a CS. The customer may not necessarily own a vehicle but can also be a family member or a user of a company fleet or car-sharing service and thus have access to the vehicle. The EV has an Original Equipment Manufacturer (OEM) Provisioning Certificate containing a unique Provisioning

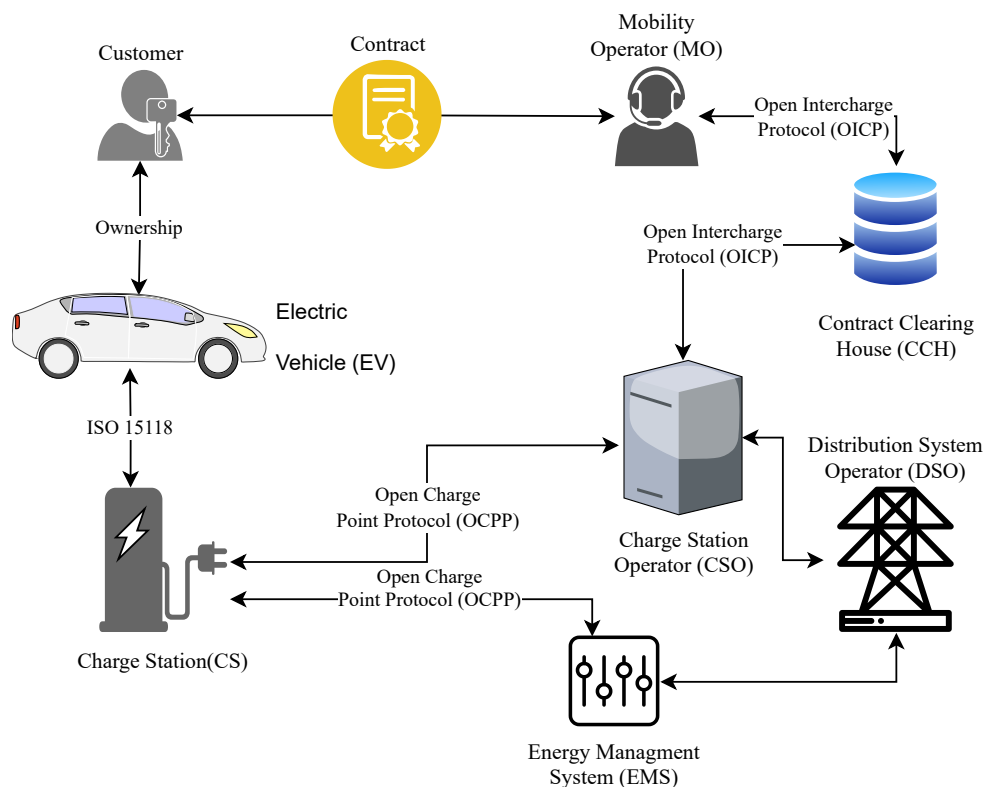


Figure 2.4.: Overview of Electric Charging Architecture

Certificate Identifier (PCID) identifying the car. This certificate is meant to be bound to the vehicle for its entire lifetime and thus is valid for around 40 years. A replacement of this certificate is only an exception and should not occur under normal circumstances. The communication unit responsible for the ISO 15118 communication in an EV is called Electric Vehicle Communication Controller (EVCC). Each EVCC has a unique Electric Vehicle Communication Controller Identifier (EVCCID).

The customer has a contract with an Mobility Operator (MO) that sells energy to the customer. Based on this contract, a Contract Certificate is issued in the context of ISO 15118, valid for up to two years. Thus, a regular certificate update is necessary. This certificate includes the E-Mobility Account Identifier (EMAID), an identifier of the contract between both MO and the customer, and typically serves as a uniquely identifies a customer. The

---

EMAID comprises a two-letter country code, the MO's ID consisting of three alphanumeric values, the individual contract identifier of nine digits and letters, and a checksum of one digit. Furthermore, the MO can offer Value Added Service (VAS), but today, only a negligible number have adopted this option. Energy providers sometimes take the MO role, but an OEM, a CSO, or another company can also take the role of an MO if they have contracts allowing them to sell energy indirectly.

Further essential parties in the context of EV charging are Charge Station Operators (CSOs) and their Charge Stations (CSs). Every CS is equipped with a Supply Equipment Communication Controller (SECC) to communicate with the vehicle during the charging process and control this process at the CS. Each SECC has a certificate containing a unique Charge Point Identifier (CPID) [90], also called Electric Vehicle Supply Equipment Identifier (EVSEID) [142], to authenticate. CSs are deployed, managed, and maintained by the CSO. Often, CSOs are energy suppliers, but new actors have also entered this market.

To coordinate the accounting and billing of charging processes between CSOs and MOs, ISO 15118 introduces the role of Contract Clearing House (CCH). Its task is to exchange authorization information between CSOs and MOs so that only legitimate customers can start a charging process. Moreover, the CCH forwards billing information and maps customer contract certificates to the responsible MO. This process is often called roaming.

The authentication procedures supported by ISO 15118 are Plug-and-Charge (PnC) or External Identification Means (EIM). The EIM can be implemented with an RFID card, mobile app, or other authentication methods, performed out-of-band at the authorization step of the ISO 15118 communication. PnC is the native authentication method based on a challenge-response procedure to validate the X.509v3 contract certificate. The contract certificate and all the previously described certificates (OEM, Electric Vehicle Supply Equipment (EVSE), etc.) are issued by a complex Public Key Infrastructure (PKI) with a deep certificate hierarchy specified in [90].

Before the ISO 15118 via the charging cable can start, a connection needs to be set up. In the first step, the cable is plugged, and the cable is locked (cf. IEC 61851). Via the control pilot wire, a PLC connection is established using HomePlug Green PHY (HPGP), a special PLC standard that, e.g., leaves out security functionalities [91]. Once this connection is established, CS and EV form an IPv6 network, where an initial service discovery process via UDP takes place to initialize the TCP connection that is used for the actual charging communication [90]. During recovery, both decide if the communication uses a unilateral TLS or plaintext.



---

The communication protocol ISO 15118 [90] on the application layer consists of a specified order of request and response message pairs. The messages have a given XML/EXI-based schema. The protocol can be divided into four phases: Communication setup, identification authentication authorization sequence, charge control re-scheduling sequence, and end of charging process. These abstract communication phases are displayed in Figure 2.5. In the first phase, the TLS connection is established.

Furthermore, the protocol version, an authentication method, charge parameters (type of charging, battery status, etc.), and a charging schedule are exchanged. The identification authentication authorization sequence either waits for the external authentication or performs the PnC authentication process. Next, in the charge control re-scheduling sequence, the vehicle is charging. In this phase, the vehicle can stop or pause the charging process at any time, or the CS can change the charging parameters, e.g., in case of a critical grid situation. The final stage is the end of the charging process, which stops the charging, transmits the final meter values for the billing process, and unlocks the cable. A detailed description is given in the privacy analysis in Section 7.3.3.

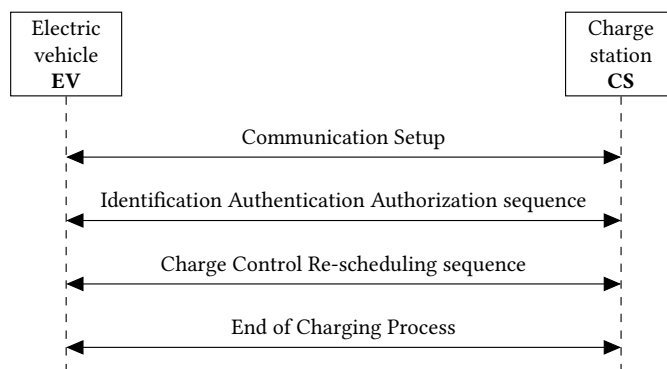


Figure 2.5.: Abstract ISO 15118 Protocol

The latest iteration of the network and application layer specification, ISO 15118-20 [88], introduces a lot of new capabilities. Notably, it supports bidirectional power transfer and wireless charging, marking a significant advancement in the domain. Furthermore, the security concept has been revised, including pivotal enhancements such as the mandatory adoption of TLS and seamless integration with Trusted Platform Module (TPM) 2.0 [211]. These pivotal updates collectively bolster the specification's functionality and security.

---

### 2.3.3. Open Charge Point Protocol (OCPP)

The most widely used communication standard for the exchange of data between CSs and the backend system controlling CSO is OCPP [140, 141], specified by the Open Charge Alliance [149, 212]. Today the most common version in commercial products is Open Charge Point Protocol (OCPP) 1.6. OCPP 1.6 and before did not require any cybersecurity mechanism but allowed all data transmission to be unauthorized and unencrypted. OCPP 1.6 allows network-level security and alternatively presents the possibility of implementing OCPP via TLS with an HTTP basic authentication scheme for CSs and possibly a certificate authentication for CSOs. Only the most recent version, OCPP 2.0.1[141], defines a basic security level by introducing TLS.

The OCPP communication scheme is a simple request-response state machine. Both CS and CSO can send requests, and the respective receiver must answer them. One core element of OCPP communication is the request for authorization of a charging EV. This request contains a unique EMAID identifier that can be provided, e.g., with an RFID token or via ISO 15118. The central backend server of the CSO confirms this request so a charging process can be initialized. Additionally, the CS can store the validity value in a so-called authorization list to authorize these identifiers in the future without requesting the backend. This list can be synchronized with the central system of the CSO. Moreover, CSs are able to notify the CSO of its current status, or the backend system may request status information. For the accounting of a charging session, the CS transmits the transaction ID, meter values, and other charging parameters to the CSO server.

CSs can also receive energy tariffs or grid signals from the CSO or a local energy management system. Furthermore, OCPP allows the maximum power to be set for a charging process or a charging schedule. This way, a smart charging process can be achieved that may reduce the cost based on power grid forecasts and ensure a stable grid. OCPP also defines messages for remote configuration CSs, such as maintenance, troubleshooting, or patching. Finally, OCPP can also be used to send arbitrary data packages.

### 2.3.4. Open Intercharge Protocol (OICP)

The Open Intercharge Protocol (OICP) [142] is an interoperable communication standard developed in 2012 by Hubeject. More than one thousand companies use OICP in 43 countries [142]. The study [149] also emphasizes its importance. The specification defines the functionality of the CCH (or E-Mobility Clearing House (EMOCH) in ISO 15118 [89]). The CCH enables communication between CSOs and MOs so customers can charge a

---

vehicle at CSs without having a contract with the corresponding CSO. The CSO can use the CCH to communicate with the customer's MO to get paid for the consumed energy. Such a service is typically a roaming service.

Implementing the OICP specification allows CSOs to authorize a charging process with the corresponding MO either directly or remotely by relaying the charging session authorization requests. This request contains the EMAID, which includes an identifier of the MO. Based on this identifier, the CCH determines the contract between the CSO and MO and forwards this information to the MO. The CSO is also informed if no contract exists. The MO then validates if the contract with the customer related to the EMAID is still valid and authorizes the charging. OICP also describes the transmission of the Charge Detail Record (CDR), which transmits the collected information during a charging process. For example, the CDR includes the EMAID, EVSEID, timestamps, and the amount of charged energy. The CDR accounts for the charging session and is archived by the CCH.

Additionally, OICP can be used to place reservations for CSs. Vehicle and status information can be uploaded and downloaded via OICP. Furthermore, the CCH can store authentication data to approve a charging session in case the MO is offline.



---

## 3. System Model

---

After the last chapter described the general technologies in modern electric vehicles, this chapter explains what a vehicle network looks like. A special focus is set on communication technologies and protocols as well as the different communication units, so-called ECUs. For every entity, a general description of its tasks is given. This automotive network architecture serves as a reference architecture in the following chapters. Next to the vehicle architecture, the reference architecture of the complex charging infrastructure is given.

Furthermore, an attacker model is described, containing relevant attackers for automotive use cases. This model includes the relevant attackers for the security solutions presented in this work. The attacker model consists of three types of attackers: the *Local Attacker*, the *Advanced Remote Attacker*, and the *Internal Attacker*.

### 3.1. Vehicular Network

The network topology of modern electric vehicles has changed due to the introduction of advanced communication standards, but also the demand to integrate additional and more advanced sensors. Modern automotive E/E architectures can be categorized according to their topology. A traditional E/E architecture based on gateways due to bandwidth limitations has been replaced by domain-based or centralized architectures [44]. Furthermore, E/E architecture differs between OEMs due to different design principles. Thus, adapting an existing architecture would narrow to this manufacturer. For this reason, an abstract E/E automotive architecture has been generated from different publicly available resources of automotive OEMs [160, 26, 195] and a survey on different vehicle networks by Miller and Valasek [127]. Figure 3.1 illustrates our generic automotive E/E we developed based on the various publicly available architectures. It is a domain-based E/E architecture, which means the vehicle has several networks divided by the functionality

and the safety level required by the functions. This architectural framework contains the internal network connections of a vehicle with various ECUs and sensors, as well as external entities directly interacting with the car and the communication channels between all these components.

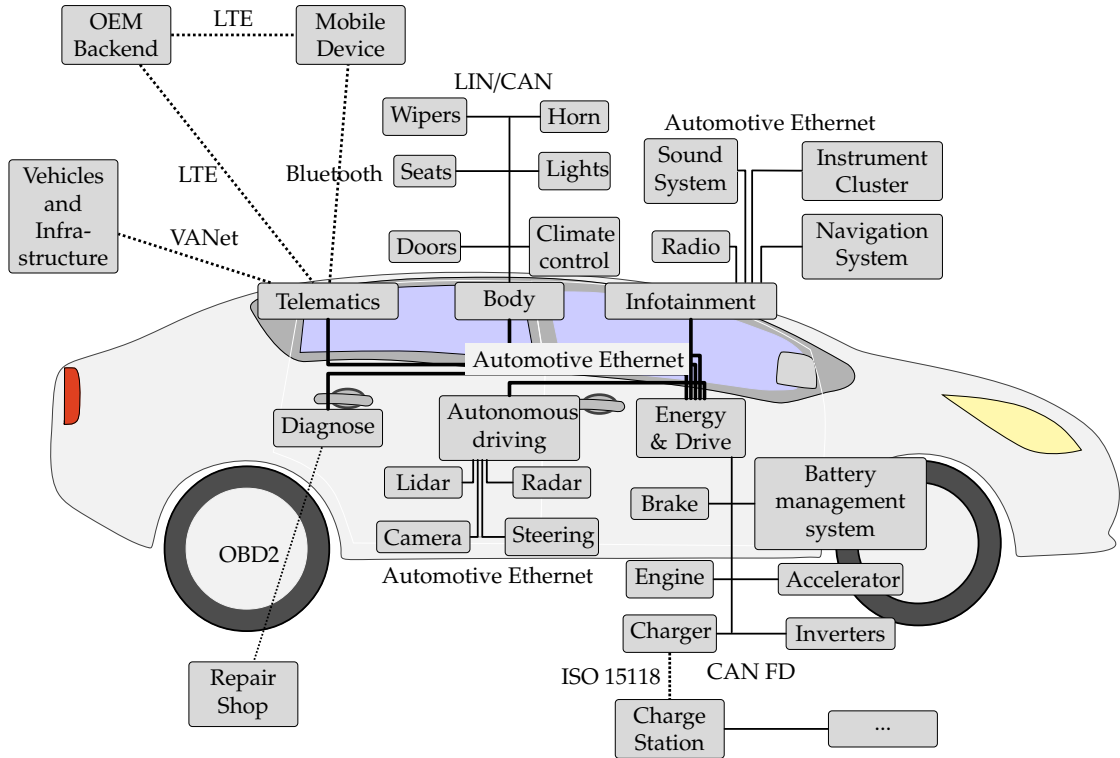


Figure 3.1.: Example Network Architecture of Modern Electric Vehicle

The reference architecture comprises five domains: drive and energy, Advanced Driver-Assistance Systems (ADAS), infotainment, telematics, body, and diagnostics. All domains are interconnected with an Automotive Ethernet backbone via a switch integrated into the energy and drive domain controller.

The energy and drive domain combines the traditional domain of driving used to control the engine and exhaust system with the energy system that includes the high-voltage battery, charging, and inverters. The domain contains ECUs that control the engine, brakes (including braking assistance like Electronic Stability Program (ESC) and Anti-lock Braking System (ABS)), exhaust cleaning system (in case it is not fully electric), the battery

---

management system, inverters that provide energy to the electric engine, accelerator, and brake paddle. Internally, the connection between these ECUs is a CAN FD bus. Furthermore, this bus system is responsible for charging the battery, so the charger ECU has a PLC connection to communicate with a charging station using ISO 15118 [88]. This includes authorization of the vehicle with PnC. Next to charging, the thermal management of the battery is also a task supervised in the drive and energy domain. The energy domain is exclusive to electric vehicles and consists of the battery, the associated battery management system, and the charge controller. The latter establishes a connection between the car and a charging station to charge the battery, e.g., via PLC and a PnC standard such as ISO 15118 [88]. Attached to the CS is the complete charging infrastructure presented in Section 2.3.

One of the most modern functionalities of vehicles is automated or autonomous driving functionalities, which are gathered in the second critical domain of safety. The ADAS system is responsible for driving decisions, has high computational power, and is connected to multiple sensors to surveil the vehicle's surroundings. These sensors are mainly cameras, radar, or lidar systems. The functionality of the ADAS ranges from automated parking over lane keeping and emergency braking to fully automated driving in the near future. Internally, the autonomous driving domain demands high bandwidth and thus uses Automotive Ethernet or CAN FD for internal communication. Furthermore, the domain controls the vehicle's steering and braking system. Since braking is part of the drive domain, these commands must be transferred via inter-domain communication. The gateway of the autonomous driving domain contains an Automotive Ethernet switch connecting all components of this domain.

Additionally, the infotainment domain in every vehicle is responsible for displaying information to the driver, e.g., speed, temperature, possible faults, and battery status. This information is typically displayed in the instrument cluster. Next to this fundamental information about the vehicle, this domain also includes the navigation system as well as entertainment functionalities, e.g., sound systems, radio, or applications like Apple CarPlay or Android Auto. It also has access ports for different user devices, like USB or AUX. Furthermore, the infotainment system allows the vehicle's configuration; this can be the temperature control, the adaptive suspension settings, the automated lighting, or the behavior of the door locks. All these configurations impact other domains and thus are communicated via the Automotive Ethernet backbone. Moreover, navigation or different infotainment system applications require internet access, e.g., to get current traffic information. For this purpose, a connection to the telematics domain is necessary. Internally, all components are connected via Automotive Ethernet with a switch integrated into the infotainment gateway.

---

The telematics domain is responsible for most of the external communication of a vehicle. The typical implementation is a TCU. The TCU is a wireless controller connected to the Automotive Ethernet. It supports different long- or short-range communication interfaces, including cellular communications, VANet for Vehicle-to-Everything (V2X) communication, and Bluetooth. The cellular communication implements the emergency call of the vehicle as well as a connection to the OEM backend for updates, traffic information, or other infotainment functions (e.g., E-Mail, Spotify, ...). V2X communication [52] allows the exchange of information about road conditions, critical situations, or traffic data with other vehicles or roadside units in a local ad-hoc network. The TCU is also equipped with an Onboard-Tester. Similar to a tester of a repair shop, this tester can perform diagnostic tasks to find problems in a vehicle and perform over-the-air updates.

The body domain consists of ECUs that control all types of functions regarding the interior and chassis of vehicles that are not highly safety-relevant. This includes the lights, horn, door control for central locking of the vehicle, and wipers, but also internal functions like climate control and (semi-) automatic seat adjustment. The communication for these functions is realized internally via CAN or Local Interconnect Network (LIN). The control signals from the infotainment system are transmitted via the Automotive Ethernet backbone to the body domain.

Finally, the OBD port gives access to the vehicle's diagnostic interface. This allows repair shops to send diagnostic messages to the vehicle network, read sensor values or error codes, and update ECUs. Third-party applications that have developed OBD dongles often use this interface to allow customers to access more detailed vehicle information. Furthermore, vehicle insurance services use this to monitor customers' driving behavior [168].

## **3.2. Communication Scenarios Electric Vehicles**

The presented architecture contains many different technologies and protocols but can be summarized into three basic communication scenarios: In-vehicle communication, external communication, and diagnostics.

### **3.2.1. In-vehicle Communication**

Communication within the vehicle network between many different ECUs is necessary to implement driving functionality. Signal-based broadcast communication has dominated



---

in-vehicle communication in the past. CAN is used for this type of communication since it is cost-efficient and fulfills real-time requirements. The low bandwidth was irrelevant for the sensor and actor values like vehicle speed, temperature, pedal angle, wheel rotations, or engine values.

Introducing more sensors for automated and assisted driving requires the transfer of much more data. The same is true for connected driving features, where highly detailed maps or infotainment data needs to be transmitted. For this reason, a service-oriented interface based on Automotive Ethernet is introduced. Service-oriented communication, like SOME/IP, enables multicast communication over UDP or TCP based on Service IDs. Furthermore, unicast communication is now possible in Automotive Ethernet.

Thus, we can distinguish in-vehicle communication between unicast, multicast, inter-domain, and service-oriented communication defined in the following.

**Unicast Communication** Unicast communication refers to the transmission of messages to a single network destination. This type of communication is necessary when sending specific data to a particular recipient. For example, the video stream from a reversing camera is only sent to the head unit.

**Multicast Communication** Multicast communication is one of the most common forms of transmission in today's vehicles. Since many vehicle communications are carried out via bus systems, every ECU in a domain can receive the transmitted data. An example of multicast communication is the broadcast of the brake signal in the safety domain, which is not only received by the brakes but also by the anti-lock braking system and potentially electric seatbelt pretensioners. In multicast communication, data from one sender is transmitted to multiple receivers, which may not necessarily be within the same domain.

**Inter-Domain Communication** As domains in vehicles have overlapping functionalities, some data must be exchanged between domains. For instance, adjusting the temperature via the infotainment system requires sending the relevant data from the infotainment domain to the air conditioning ECU in the body domain. In this case, the domain controllers forward the data from one domain to another. Inter-domain communication typically occurs between two ECUs, commonly across the border of domains.

---

**Service-Oriented Communication** Another notable trend in in-vehicle communication is the shift from signal-based communication to service-based communication, where each ECU offers a set of services that other ECUs can subscribe to or request information from. The most widely adopted protocol for service-oriented communication is SOME/IP, as described in Section 2.2.1.

### 3.2.2. External Communication

Next to internal communication, a vehicle also communicates with the outside world through different channels. Modern vehicles are connected to the external world through various communication interfaces. The most common is the direct connection to the OEM backend via cellular communication. This is used for different functions, e.g., connected to a mobile app to track, unlock, or start the vehicle. Furthermore, it allows remote updates or feature unlocks as well as transmission of communication data. Additionally, direct communication of the car to the Charge Station via Power-Line Communication or WLAN for the charging communication is implemented in electric vehicles for authorization, billing, and scheduling. All these connections are direct unicast communications between the vehicle and one external entity over a TCP or UDP connection.

In addition to these communication cases, V2X allows local broadcast communication to other vehicles and roadside units, exchanging traffic or safety information. V2X is either realized with IEEE 802.11p Wi-Fi or cellular communication. Since V2X has been covered extensively in previous literature [102], it is not the focus of this work.

### 3.2.3. Diagnostics

A special communication of vehicles is diagnostics. Diagnostics information is transmitted inside the car to warn the driver of critical conditions and inform a service technician to help detect problems. Furthermore, the diagnostics functionality can be used to update, configure, or test the vehicle, e.g., for emission testing. Traditionally, a repair shop connects to the OBD-II port of a vehicle for diagnostics, but lately, remote diagnostics has also become more and more important.

Diagnostic requests are typically unicast connections but need to be routed to the corresponding ECU from the tester through domain controllers to the actual ECU.

---

## 3.3. Attacker Model

For the presented communication channels, transport safety critical information or private data can both be targets for attackers. This section describes relevant attacker models derived from real attacks or attack scenarios: *Local Attacker*, *Advanced Remote Attacker*, and *Internal Attacker*. These attack models are used for the threat and risk analyses in the following chapters. Since the approach in the next chapter describes the attack feasibility based on the difficulty of attacks, the attacker models are only needed to define an entry point for the attacker. Local Attacker

### 3.3.1. Local Attacker

The first attacker model is the *Local Attacker*, based on the first in-vehicle communication attacks presented by Kocher et al. in [106]. Kocher et al. could control many vehicle functions by injecting messages to the CAN bus, including acceleration and braking. Next to this finding, attacks on OBD dongles [59] or the introduction of tuning devices or AdBlue emulators [68] to the vehicle network allows attackers to inject messages into the vehicle network to manipulate the functionality of the vehicle either to increase performance, to reduce the emission cleaning or to cause harm to the passengers.

The attack model allows the attacker to send arbitrary messages on any in-vehicle network. Furthermore, the attacker can flip bits of messages sent by an honest, legitimate ECU. Finally, she can monitor all messages sent in the in-vehicle network so all messages can be replayed.

However, the attacker cannot access cryptographic keys and cannot use cryptographic methods. Attacks to the software and keys stored on ECUs are prevented using countermeasures, e.g., secure boot and a secure key store implementation using Hardware Security Modules (HSMs). Thus, legitimate will behave correctly in this attacker model.

### 3.3.2. Advanced Remote Attacker

Widespread attacks on different automotive OEMs have shown that the assumption from the previous attacker model about the security of ECUs is not necessarily valid. In the Jeep hack [128] the researchers could obtain control over the infotainment and finally trigger automated parking commands. Furthermore, the attacks on the Tesla Model S [137],

---

the Volkswagen infotainment system [101], BMW [27], and Mercedes-Benz [200] have pointed out the danger for ECUs.

For the *Advanced Remote Attacker*, the assumptions of the *Local Attacker* are extended by the ability to compromise ECUs, but with the restriction of not being able to access HSMs and the inability to break cryptographic primitives. Internal Attacker

### **3.3.3. Internal Attacker**

The EV charging ecosystem is intricate and involves diverse stakeholders, as detailed in Section 2.3.1. These stakeholders range from large corporations to start-ups, each with varying degrees of security measures in place. The substantial number of entities participating in the charging process significantly impacts the security and privacy of EV users.

Given these complexities, it is reasonable to consider the potential compromise of an entity within the charging infrastructure. In this context, the concept of an *Internal Attacker* pertains to an attacker with the capability to access all information stored at or transmitted to a specific charging entity. This extends to all messages sent and received by this entity, a scenario reminiscent of a Dolev-Yao attacker [47].

---

## 4. Risk Analysis of Next Generation Electric Vehicles

---

To identify potential threats to modern electric vehicles, we conducted a comprehensive Threat Analysis and Risk Assessment (TARA) of the reference architecture outlined in Section 3.1. This assessment was carried out in collaboration with experts from the industry and researchers, as detailed in the publications [208, 207]. These results have been partially produced in multiple projects with a German vehicle manufacturer and a tier-2 supplier, and both are using the results and insides of our presented methodology in their operations. Notably, my contributions include a method to identify attack paths based on orchestrating the attack feasibility rating process and designing and implementing an automated algorithm for generating attack feasibility paths, which determine the most viable attack paths for potential adversaries.

The approach we present in this report is not a standalone method. It aligns with the ongoing initiatives to standardize and regulate cybersecurity practices. For instance, the United Nations Economic Commission for Europe (UNECE) has issued regulations 155 on "Cyber Security and Cyber Security Management System" [184] and 156 on "Software Update and Software Update Management System" [185], which render cybersecurity requirements crucial for the approval of new vehicles and types. Our approach, which is specifically developed with adherence to ISO/SAE 21434, is in line with these regulations. The standards SAE J3061 [159] and ISO/SAE 21434 [95] have been developed to provide a process for conducting TARAs in compliance with the UNECE 155 regulation. Next to the TARA, ISO/PAS 5112 [94] released 2022 introduces an audit schema for the Cyber Security Management System of [184] and ISO 24089 [93] specifies requirements and recommendations for software update in correspondence to [185].

The structure of this chapter unfolds as follows: We begin by introducing the risk assessment process outlined in ISO/SAE 21434, a standard that plays a pivotal role in our TARA process. Subsequently, we assess the attack surface of the vehicle, followed by the attack

---

feasibility rating. Additionally, we elaborate on our self-developed process for determining attack paths within the vehicle network, addressing a gap not explicitly covered by the standard. Based on the identified paths and the initial ratings of individual attack steps, we determine the feasibility of the resulting attack paths. These findings ultimately guide our decision on the technologies to investigate further in the subsequent chapters.

## **4.1. The ISO/SAE 21434 Threat Analysis and Risk Assessment (TARA) Process**

The year 2021 marked the release of the international standard ISO/SAE 21434 [95] for cyber-security engineering in the automotive domain, replacing the SAE J3061 "Cyber-security Guidebook for Cyber-Physical Vehicle Systems" [159]. This standard outlines the Attack Surface Assessment with the TARA process, which involves multiple steps illustrated in Figure 4.1.

The TARA process begins with the identification of relevant assets and corresponding threat scenarios. For each identified threat scenario, the potential attack paths leading to that scenario are analyzed, and the feasibility of these attacks is rated. Concurrently, the potential impact caused by each threat scenario is assessed through an impact rating. The combination of attack feasibility and impact ratings then enables the determination of risk levels. Lastly, appropriate risk treatment measures must be suggested to address the identified risks.

The standard provides informative annexes that propose weighting criteria, categories, and matrices to support the TARA process. These criteria, along with their suggested value ranges, are optional and can be adapted as needed. It is worth noting that the ISO standard allows for using modified values or incorporating additional criteria in the context of a risk analysis that aligns with the principles outlined in ISO/SAE 21434.

## **4.2. Attack Surface Assessment**

The initial step of the TARA process, as outlined in ISO/SAE 21434 [95] (cf. Figure 4.1) and Annex G, involves the identification of assets. Following this, the standard suggests identifying threat scenarios, which are then utilized to determine attack paths and assess the attack feasibility. However, this step is not explicitly described in the standard. In light

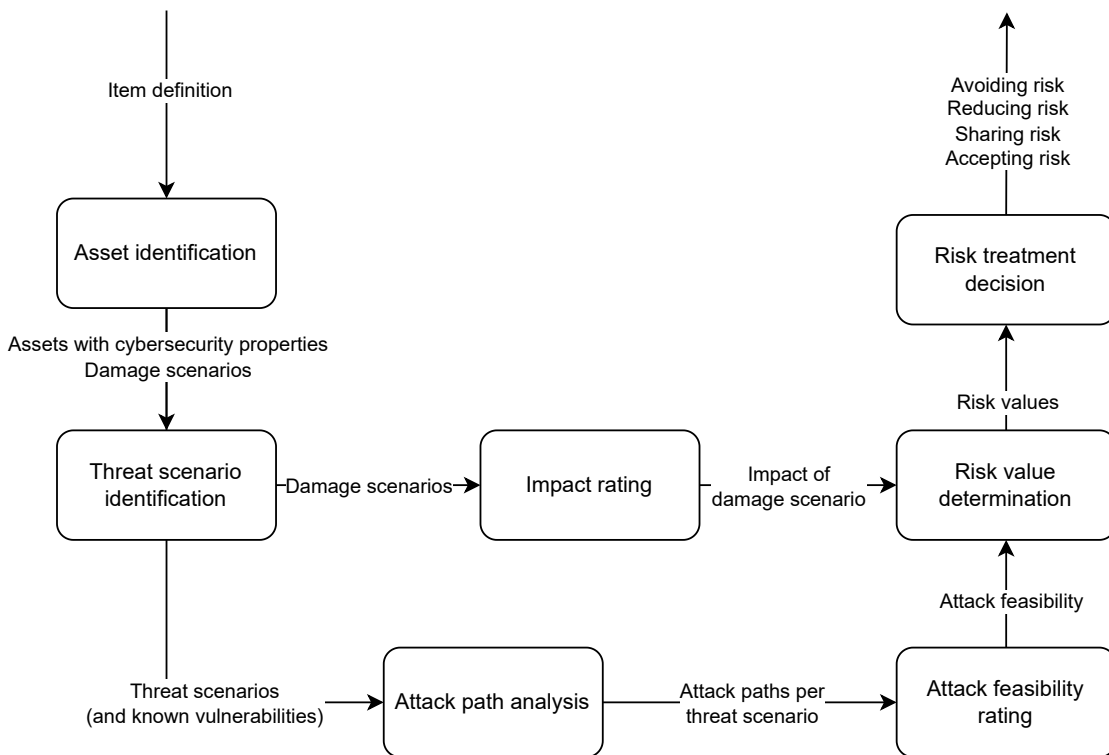


Figure 4.1.: Process to determine Risk according to [95]

of this, we propose an approach for automatically generating attack paths and assessing their feasibility based on rated basic attack building blocks.

#### 4.2.1. Asset Definition in Modern Electric Vehicles

Drawing from the reference architecture presented in Section 3.1, we have identified five categories of assets that require protection within modern electric vehicles:

1. Cryptographic keys and functionalities,
2. Wireless vehicle interfaces and communications,
3. Wired vehicle interfaces and communications,

- 
4. On-car ECUs including directly connected actors, and
  5. On-car sensors.

Each asset category can be further divided into various attack methods, considering the different technologies employed within the vehicle.

**Cryptographic keys** Cryptographic keys are integral to several security mechanisms, such as secure communication and access control. For example, symmetric keys are used in AUTOSAR's SecOC [8] to secure in-vehicle communication. In contrast, asymmetric keys are utilized in ISO 15118 PnC authentication for charging credentials [88]. In this category, we consider the feasibility of breaking cryptographic algorithms and illegitimately acquiring or modifying cryptographic keys. We distinguish between hardware and software attacks targeting keys stored within ECU memory and those stored in shielded locations such as HSMs or TPMs.

**Wireless vehicle interfaces and communications** Wireless interfaces and communication channels provide avenues for remote attacks without physical access to the vehicle. We consider the feasibility of intercepting, listening, jamming, corrupting, altering, injecting, or replaying messages transmitted through WiFi, cellular, GPS, and Bluetooth interfaces.

**Wired vehicle interfaces and communications** This category addresses attackers with physical access to the vehicle. Attackers can exploit exposed interfaces such as OBD, debug interfaces like JTAG, USB, and AUX, indirectly accessible interfaces to bus systems including CAN, CAN FD, FlexRay, and Ethernet, as well as interfaces to the environment like PLC for PnC. These attacks often serve as entry points for more sophisticated attacks.

**On-car ECUs** The ECUs present in a vehicle offer various assets that attackers may seek to compromise. We consider attacks such as vulnerability exploitation, Denial of Service (DoS) or ECU disabling, configuration changes, flashing of malicious code, and execution of malicious code (potentially with escalated privileges).

**On-car sensors** Modern vehicles are equipped with multiple sensors, including those for pedal position, steering angle, ultrasonic, LiDAR, radar, and cameras. Attackers may attempt to spoof sensor signals or manipulate sensor input to deceive the vehicle's systems.



---

## 4.2.2. Attack Feasibility Rating

Our methodology for determining the feasibility of basic attacks is based on the attack potential-based approach presented in ISO/IEC 18045 [87]. This approach, also recommended in ISO/SAE 21434 [95], introduces five dimensions: Elapsed Time, Specialist Expertise, Knowledge of the Item or Component, Window of Opportunity, and Equipment.

The *Elapsed Time* dimension characterizes the time required to prepare and execute an attack, which can range from less than a week to more than three years.

The *Specialist Expertise* dimension describes the attacker's level of skill and knowledge. It is categorized as a layman (no particular expertise), proficient (familiar with the target's behavior), expert (possessing deep knowledge in a specific technique such as cryptanalysis), or multiple experts (with expertise in different fields).

The *Knowledge of the Item or Component* dimension indicates the difficulty of the attack, classified as public, restricted, confidential, or strictly confidential information required to execute the attack.

The *Window of Opportunity* dimension represents the timeframe during which an attacker can carry out an attack, primarily constrained by target accessibility. Basic attacks only consider immediate opportunities without pre-limiting conditions. For example, sending a message on a bus depends on access to the bus, whether through physical access or an ECU. However, once the precondition is met, the window of opportunity is unlimited. This dimension includes the categories unlimited, easy, moderate, and difficult.

The *Equipment* dimension accounts for the necessary tools and resources that an attacker needs to successfully execute an attack, classified as standard, specialized, bespoke, or multiple bespoke.

Following the recommended value scale from Annex I of ISO/SAE 21434 [95], numerical values are assigned to the quantitative values of the previously mentioned dimensions.

The combination of all ratings, obtained by summing up the numerical values, determines the attack feasibility rating of a basic attack. The resulting attack feasibility rating can fall into the categories: Very Low, Low, Medium, or High. Annex I of ISO/SAE 21434 [95] provides a recommended value scale for transforming the numerical values back into qualitative values. Our rating for different basic attacks is listed in Tables 4.1, 4.2, 4.3, 4.3, 4.4, 4.5, and 4.6, which includes all basic attacks against the vehicle architecture introduced in Section 3.1.

Table 4.1.: Attack Feasibility Rating for Attacks Against Cryptographic Keys

Id	Asset (attack)	Elapsed time	Specialist expertise	Knowledge of item/ component	Window of opportunity	Equipment	Attack feasibility
1.1	Keys (illegal acquisition, modification or breaking): Extract from HSM (Softwarebug)	<3 years	Expert	Restricted	Unlimited	Specialized	Low
1.2	Keys (illegal acquisition, modification or breaking): Extract from HSM (Hardwareattack)	<3 years	Multiple experts	Confidential	Difficult	Bespoke	Very Low
1.3	Keys (illegal acquisition, modification or breaking): Extract from TPM (Softwarebug)	>3 years	Expert	Restricted	Unlimited	Specialized	Very Low
1.4	Keys (illegal acquisition, modification or breaking): Extract from TPM (Hardwareattack)	>3 years	Multiple experts	Confidential	Difficult	Bespoke	Very Low
1.5	Keys (illegal acquisition, modification or breaking): Extract from Firmware (Software)	<1 month	Proficient	Confidential	Unlimited	Specialized	Medium
1.6	Keys (illegal acquisition, modification or breaking): Break Cryptographic algorithm (min. AES-128/ RSA 2048/ ECC 256)	>3 years	Expert	Public	Unlimited	Standard	Very Low
1.7	Keys (illegal acquisition, modification or breaking): Extract from Firmware (Hardware)	<3 years	Expert	Confidential	Difficult	Bespoke	Very Low
1.8	Keys (forge): Brute Force SecOC	<1 week	Proficient	Restricted	Difficult	Standard	Medium

Table 4.2.: Attack Feasibility Rating for Attacks Against Wireless Communication Interfaces

Id	Asset (attack)	Elapsed time	Specialist expertise	Knowledge of item/component	Window of opportunity	Equipment	Attack feasibility
2.1	On-car wireless interfaces (access): Bluetooth	<1 week	Proficient	Public	Easy	Standard	High
2.2	On-car wireless interfaces (access): Cellular	<1 week	Proficient	Public	Unlimited	Specialized	High
2.3	On-car wireless interfaces (access): WiFi	<1 week	Proficient	Public	Easy	Standard	High
2.4	On-car wireless interfaces (access): GPS	<1 week	Proficient	Public	Unlimited	Specialized	High
2.5	Wireless Com. (jamming): GPS	<1 week	Layman	Public	Easy	Specialized	High
2.6	Wireless Com. (jamming): WiFi (IEEE 802.11p)	<1 week	Layman	Public	Easy	Standard	High
2.7	Wireless Com. (jamming): Cellular (LTE/5G)	<1 week	Layman	Public	Easy	Specialized	High
2.8	Wireless Com. (corrupt / fake msg and info): WiFi (IEEE 802.11p)	<1 week	Proficient	Public	Easy	Standard	High
2.9	Wireless Com. (corrupt / fake msg and info): Cellular (LTE/5G)	<1 month	Proficient	Public	Easy	Specialized	High
2.10	Wireless Com. (corrupt / fake msg and info): GPS (spoofing)	<1 month	Proficient	Public	Easy	Specialized	High
2.11	Wireless Com. (corrupt / fake msg and info): Connected Car (via Cellular)	<1 week	Proficient	Public	Unlimited	Standard	High
2.12	Wireless Com. (listen): WiFi (IEEE 802.11p)	<1 week	Proficient	Public	Easy	Standard	High
2.13	Wireless Com. (listen): Cellular (LTE/5G)	<1 month	Proficient	Public	Easy	Specialized	High
2.14	Wireless Com. (listen): Bluetooth (BLE)	<1 month	Proficient	Public	Easy	Specialized	High
2.15	Wireless Com. (intercept, alter, inject, replay): WiFi (IEEE 802.11p)	<1 week	Proficient	Public	Easy	Standard	High
2.16	Wireless Com. (intercept, alter, inject, replay): Cellular (LTE/5G)	<1 month	Proficient	Public	Easy	Specialized	High
2.17	Wireless Com. (intercept, alter, inject, replay): Bluetooth (BLE)	<1 month	Proficient	Public	Easy	Specialized	High

Table 4.3.: Attack Feasibility Rating of Attacks Against Wired Communication Interfaces

Id	Asset (attack)	Elapsed time	Specialist expertise	Knowledge of item/component	Window of opportunity	Equipment	Attack feasibility
3.1	On-car interfaces (access – physical tampering): CAN/CAN FD, Automotive Ethernet	<1 week	Proficient	Restricted	Moderate	Standard	High
3.2	On-car interfaces (access – physical tampering): FlexRay	<1 week	Proficient	Restricted	Moderate	Specialized	Medium
3.3	On-car interfaces (access – physical tampering): Debug interfaces (e.g. JTAG) (for easy to access components)	<1 month	Expert	Restricted	Moderate	Specialized	Medium
3.4	On-car interfaces (access – physical tampering): Debug interfaces (e.g. JTAG) (for components with difficult access e.g. HV Battery)	<1 month	Expert	Restricted	Difficult	Specialized	Low
3.5	On-car interfaces (physical tampering): OBD	<1 week	Layman	Public	Easy	Standard	High
3.6	On-car interfaces (physical tampering): PLC	<1 month	Proficient	Public	Easy	Specialized	High
3.7	On-car user hardware interfaces (access): USB	<1 week	Layman	Public	Easy	Standard	High
3.8	On-car user hardware interfaces (access): Aux	<1 week	Layman	Public	Easy	Standard	High
3.9	Wired Communications (corrupt / fake msg and info): USB	<1 week	Proficient	Public	Easy	Standard	High
3.10	Wired Communications (corrupt / fake msg and info): AUX	<1 week	Proficient	Public	Easy	Specialized	High
3.11	Wired Com. (intercept, alter, inject, replay): USB	<1 week	Proficient	Public	Easy	Standard	High
3.12	Wired Com. (intercept, alter, inject, replay): AUX	<1 week	Proficient	Public	Easy	Specialized	High
3.13	Wired Com. (intercept, alter, inject, replay): PLC	<1 week	Proficient	Public	Easy	Specialized	High

Table 4.3.: Attack Feasibility Rating (Continued)

Id	Asset (attack)	Elapsed time	Specialist expertise	Knowledge of item/component	Window of opportunity	Equipment	Attack feasibility
Wired On-Car Interfaces and Communications							
3.14	On-car Communications (listen + understand): PLC	<1 month	Proficient	Public	Easy	Standard	High
3.15	Wired/Wireless Com. (spoof): External test and diagnostic equipment	<1 week	Layman	Public	Unlimited	Specialized	High
3.16	On-car Communications (disable or Denial of Service): CAN/CAN FD, FlexRay, Automotive Ethernet	<1 week	Proficient	Public	Unlimited	Standard	High
3.17	On-car Communications (listen): CAN/CAN FD, FlexRay, Automotive Ethernet	<1 month	Proficient	Public	Unlimited	Standard	High
3.18	On-car Communications (intercept): CAN/CAN FD, FlexRay, Automotive Ethernet	<1 week	Proficient	Public	Unlimited	Standard	High
3.19	On-car Communications (replay): CAN/CAN FD, FlexRay, Automotive Ethernet	<1 week	Proficient	Public	Unlimited	Standard	High
3.20	On-car Communications (inject): CAN/CAN FD, FlexRay, Automotive Ethernet	<1 week	Proficient	Public	Unlimited	Standard	High
3.21	On-car Communications (alter): CAN/CAN FD, FlexRay	<1 week	Expert	Public	Unlimited	Specialized	High
3.22	On-car Communications (alter): Automotive Ethernet	<1 week	Proficient	Public	Unlimited	Standard	High

Table 4.4.: Attack Feasibility Rating of Attacks Against ECUs

Id	Asset (attack)	Elapsed time	Specialist expertise	Knowledge of item/component	Window of opportunity	Equipment	Attack feasibility
4.1	On-car ECU (exploit vuln. or impl. error to access ECU): ECU with external interface (wireless (Cellular/BLE/Wifi))	<6 months	Proficient	Restricted	Unlimited	Specialized	Medium
4.2	On-car ECU (exploit vuln. or impl. error to access ECU): ECU with external interface (wired (OBD/PLC/USB/AUX))	<6 months	Proficient	Restricted	Unlimited	Specialized	Medium
4.3	On-car ECU (exploit vuln. or impl. error to access ECU): ECU with internal interface (wired (CAN/CAN FD/FlexRay/Ethernet))	<6 months	Proficient	Restricted	Unlimited	Standard	High
4.4	On-car ECU (exploit vuln. or impl. error to access ECU): ECU with debug interface (wired (UART/JTAG/...))	<6 months	Proficient	Restricted	Unlimited	Specialized	Medium
4.5	On-car ECU (exploit vuln. or impl. error to access ECU): XCP (via CAN/CAN FD)	<6 months	Proficient	Restricted	Unlimited	Standard	High
4.6	On-car ECU (disable or Denial of Service): Resource exhaustion of regular ECU	<1 week	Proficient	Restricted	Unlimited	Standard	High
4.7	On-car ECU (disable or Denial of Service): Shutdown/Halt	<1 month	Proficient	Restricted	Unlimited	Standard	High
4.8	On-car ECU (disable or Denial of Service): Resource exhaustion of High Performance ECU	<1 week	Expert	Restricted	Unlimited	Specialized	High

Table 4.5.: Attack Feasibility Rating of Attacks Against ECUs (Continued)

Id	Asset (attack)	Elapsed time	Specialist expertise	Knowledge of item/component	Window of opportunity	Equipment	Attack feasibility
4.9	On-car ECU (configuration change): Remote	<1 month	Proficient	Restricted	Unlimited	Standard	High
4.10	On-car ECU (configuration change): Physical	<1 month	Proficient	Restricted	Moderate	Specialized	Medium
4.11	On-car ECU (remote malware flash): No integrity measures	<1 week	Proficient	Restricted	Unlimited	Standard	High
4.12	On-car ECU (remote malware flash): With integrity measures	<3 years	Proficient	Restricted	Unlimited	Standard	Medium
4.13	On-car ECU (flash via physical access): ECU without integrity measures external flash	<1 week	Proficient	Restricted	Moderate	Specialized	High
4.14	On-car ECU (flash via physical access): ECU with integrity measures (e.g., secure boot or measured boot)	<3 years	Proficient	Restricted	Moderate	Specialized	Low
4.15	On-car ECU (flash via physical access): ECU without integrity measures with embedded flash	<6 months	Expert	Restricted	Moderate	Bespoke	Low
4.16	On-car ECU (exploit for priv. Escalation)	<6 months	Proficient	Restricted	Unlimited	Standard	High
4.17	On-car ECU (execute Code/Commands)	<1 month	Proficient	Public	Unlimited	Standard	High
4.18	On-car ECU: Access to Replacement Parts	<1 week	Layman	Public	Unlimited	Standard	High

Table 4.6.: Attack Feasibility Rating of Attacks Against Sensors

Id	Asset (attack)	Elapsed time	Specialist expertise	Knowledge of item/component	Window of opportunity	Equipment	Attack feasibility
5.1	On-car Sensors (spoof of sensor Signal): Brake pedal position, Throttle pedal position, Steering angle sensor	<1 week	Proficient	Restricted	Moderate	Specialized	Medium
5.2	On-car Sensors (spoof of sensor Signal): Ultrasonic, Lidar, Radar Sensor	<1 week	Proficient	Restricted	Moderate	Specialized	Medium
5.3	On-car Sensors (spoof of sensor Signal): Rear view camera, Stereo front camera	<1 week	Proficient	Restricted	Moderate	Specialized	Medium
5.4	On-car Sensors (disable or Denial of Service): Brake pedal position, Throttle pedal position, Steering angle sensor	<1 week	Proficient	Restricted	Moderate	Standard	High
5.5	On-car Sensors (disable or Denial of Service): Ultrasonic, Lidar, Radar Sensor	<1 week	Layman	Public	Easy	Standard	High
5.6	On-car Sensors (disable or Denial of Service): Rear view camera, Stereo front camera	<1 week	Layman	Public	Easy	Standard	High
5.7	On-car Sensors (external manipulation of sensor input): Brake pedal position, Throttle pedal position, Steering angle sensor	<1 week	Proficient	Restricted	Moderate	Specialized	Medium
5.8	On-car Sensors (external manipulation of sensor input): Ultrasonic, Lidar, Radar Sensor	<1 week	Proficient	Restricted	Moderate	Specialized	Medium
5.9	On-car Sensors (external manipulation of sensor input): Rear view camera, Stereo front camera	<1 week	Proficient	Restricted	Moderate	Specialized	Medium



---

### 4.2.3. Discussion on Different Path Calculation Methods

The generation of attack paths is a crucial step in the TARA process. We need to establish a function that determines attack feasibility based on the basic attack steps to automatically generate the most feasible attack paths. We present five different approaches and discuss their advantages, leading to the description of the chosen process.

**Sum:** This approach involves summing up all feasibility values along the path per category. It is a simple and intuitive method that facilitates the comparison of paths of equal length. However, comparing attack paths of different lengths can be challenging since a long path with low difficulty at some point will reach the highest difficulty step. Additionally, projecting the resulting values back to the attack feasibility categories defined by ISO/SAE 21434 is difficult using this approach.

**Average:** The average approach calculates the average feasibility value per category along the path. This addresses the disadvantage of the sum approach, as the values obtained can be easily transferred to the rating choices per category. However, the average function has a balancing property, which means that an attack path with one difficult step will gain feasibility as more easy attack steps are added. Due to this property, we did not choose this calculation method.

**Maximum:** In this approach, we consider the maximum feasibility value for each step of the attack path per category. This ensures that the difficulty cannot be reduced along the path. However, it does not adequately represent the difficulty of longer attack paths compared to shorter paths.

**Hybrid:** In the hybrid approach, we combine different methods based on the category. We use the sum approach for the *Elapsed Time* category since it is intuitive, to sum up the time necessary for attack preparation and execution. We choose the maximum approach for the categories of *Specialist Expertise*, *Knowledge of the Item/Component*, and *Equipment* since these values cannot be added together. An attacker either needs expert knowledge during the attack or not, and the same goes for equipment and knowledge. The *Window of Opportunity* category is a special case since it characterizes the attacker's opportunity to attack the vehicle. For example, an attack via a cellular connection can be executed

---

anytime with an Internet connection (if no restrictions exist). In contrast, the execution of a CAN command can only be done locally, significantly reducing the window of opportunity. We believe only the first attack step is essential to identify the window of opportunity.

Considering the drawbacks of the earlier approaches, we have decided to proceed with the hybrid model for the exemplary application of the reference architecture described in Section 3.1.

### 4.3. Automated Attack Path Generation

In order to derive attack paths efficiently and with fewer errors compared to manual processes, we have developed a partially automated method that allows for quick adaptation and faster results. This automated approach consists of several steps: Directed Weighted Graph Generation, Attack Path Extraction, Threat Path Extraction, and Most Feasible Path Calculation. The overall process is illustrated in Figure 4.2, and each step is explained in detail below.

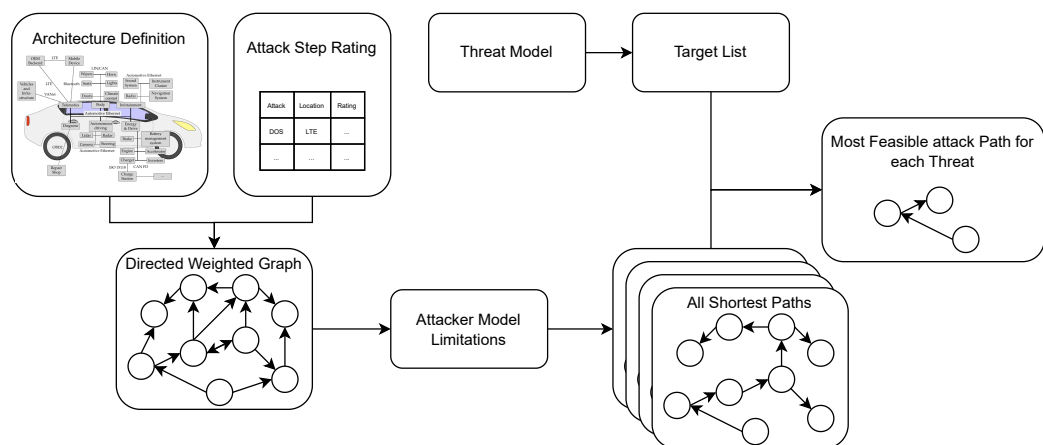


Figure 4.2.: Automated Method

The input for the automated process is a manually created system model based on the possible attack steps described in Section 4.2. For each attack step, we define preconditions and the location of the attack within the vehicle model (see Section 3.1). For example, an attacker may require a physical connection or have compromised an ECU connected

---

to a specific bus in order to send a CAN-Bus message. Each node in the system model is based on an attack step (e.g., "17.2 On-car ECU (configuration change): Physical") and a location (e.g., "Headunit").

### 4.3.1. Example

To illustrate the following description of our automated process, let's consider a simple example with three attack types:  $a_1$ ,  $a_2$ , and  $a_3$ . The attack step  $a_1$  is rated as (" $<1$  week", "Expert", "Public", "Standard"),  $a_2$  as (" $<1$  year", "Expert", "Public", "Bespoke"), and  $a_3$  as (" $<6$  months", "Layman", "Public", "Specialized"). In the simplified example the model only contains a single ECU with a bus connection. In this example, an attacker can perform either  $a_1$  or  $a_2$  on the bus before being able to perform  $a_3$  on the ECU. Our analysis focuses on the attack goal of performing  $a_3$  on the ECU.

### 4.3.2. Directed Weighted Graph Generation

To generate the directed weighted graph for the following path generation steps we designed a process which is described in the following. The pseudo-code of the is given in Algorithm 1. In this step of the attack path generation, our automated method generates a directed graph by connecting the attack steps according to the defined preconditions and locations. In lines 2-14, we add a node to the attack graph for every attack, introduced in Section 4.2.2, at every possible location. Next, in lines 15-30, we introduce the edges to the attack graph. This forms a directed attack graph representing all possible attack paths in our vehicle model.

### 4.3.3. Example

After performing this algorithm in our example, is used to determine we get the attack graph with three nodes:  $a_1@BUS$  (" $<1$  week", "Expert", "Public", "Standard"),  $a_2@BUS$  (" $<1$  year", "Expert", "Public", "Bespoke"), and  $a_3@ECU$  (" $<6$  months", "Layman", "Public", "Specialized"). Both  $a_1@BUS$  and  $a_2@BUS$  have edges to  $a_3@ECU$  with the weight (" $<6$  months", "Layman", "Public", "Specialized").

---

**Algorithm 1** Initialize attack graph

---

```
1:  $G \leftarrow \{\}$ 
2: for all node  $\in$  VehicleModel do
3:   for all attack  $\in$  AttackSteps do
4:     if node  $\in$  attack.locations then
5:        $n \leftarrow \text{node}(\text{attack.id}@node.\text{location})$ 
6:        $n.time = \text{attack.time}$ 
7:        $n.experience = \text{attack.experience}$ 
8:        $n.knowledge = \text{attack.knowledge}$ 
9:        $n.opportunity = \text{attack.opportunity}$ 
10:       $n.equipment = \text{attack.equipment}$ 
11:       $G.nodes \leftarrow G.nodes \cup \{n\}$ 
12:    end if
13:  end for
14: end for
15: for all  $n1 \in G$  do
16:   for all  $n2 \in G$  do
17:     if  $n1.location == n2.location \wedge n1.location \in n2.location.edges$  then
18:       if  $n1.attack \in n2.attack.prestep$  then
19:          $G. = \text{attack.equipment}$ 
20:          $e \leftarrow \text{edge}(n1,n2)$ 
21:          $G.edges \leftarrow G.edges \cup \{e\}$ 
22:          $e.time = n2.time$ 
23:          $e.experience = n2.experience$ 
24:          $e.knowledge = n2.knowledge$ 
25:          $e.opportunity = n2.opportunity$ 
26:          $e.equipment = n2.equipment$ 
27:       end if
28:     end if
29:   end for
30: end for
31: return  $G$ 
```

---

---

#### 4.3.4. Attack Path Extraction

To prepare the extraction of attack paths in the next step, we prepare the calculation of shortest paths in the attack graph using Algorithm 2. In line 2 we introduce a virtual starting node ( $vStart$ ) to the graph and connect this node in lines 3-12 to possible starting nodes. In this example, we use every node without a previous attack step as a starting node (line 4). At this stage, it is possible to apply restrictions to the attacker, such as considering only remote attacks by not connecting physical attacks to  $vStart$ . We assign weights to all edges from the virtual starting node to the starting nodes of an attacker. The weights are the attack feasibility ratings of the starting nodes of an attacker. We then assign weights to all edges in the graph based on the attack feasibility of the corresponding attack steps (lines 7-11).

Finally, we calculate the shortest loop-free paths in the directed attack graph starting from  $vStart$ . The BellmanFord algorithm [14, 58] is used to determine all shortest paths in the attack graph starting with the virtual starting node (line 14). We modified the weight function to the "hybrid" method introduced in Section 4.2.3 taking into account the difficulty ratings for time, expertise, knowledge, opportunity, and equipment. The result is a set of shortest paths representing the different attack paths in the graph.

---

**Algorithm 2** Prepare shortest paths in attack graph

---

```
1:  $G \leftarrow$  initialize AttackGraph
2:  $G.nodes \leftarrow G.nodes \cup \{vStart\}$ 
3: for all node  $\in G.nodes$  do
4:   if  $|G^T.edges[node]| = 0 \wedge node \in AttackerModel$  then
5:      $e \leftarrow edge(vStart,node)$ 
6:      $G.edges \leftarrow G.edges \cup \{e\}$ 
7:      $e.time = n2.time$ 
8:      $e.experience = n2.experience$ 
9:      $e.knowledge = n2.knowledge$ 
10:     $e.opportunity = n2.opportunity$ 
11:     $e.equipment = n2.equipment$ 
12:   end if
13: end for
14:  $shortestPaths \leftarrow BellmanFord(G.nodes, G.edges, vStart)$ 
15: return  $shortestPaths$ 
```

---

---

### 4.3.5. Example

After preparing the attack paths the attack graph contains the node  $vStart$  with two edges to  $a_1@BUS$  where the weight is (" $<1$  week", "Expert", "Public", "Standard") and to  $a_2@ECU$  with the weight (" $<1$  year", "Expert", "Public", "Bespoke"), and  $a_3@ECU$  (" $<6$  months", "Layman", "Public", "Specialized"). Furthermore, all shortest paths are calculated from  $vStart$  to the other three nodes. The final graph is given in Figure 4.3.

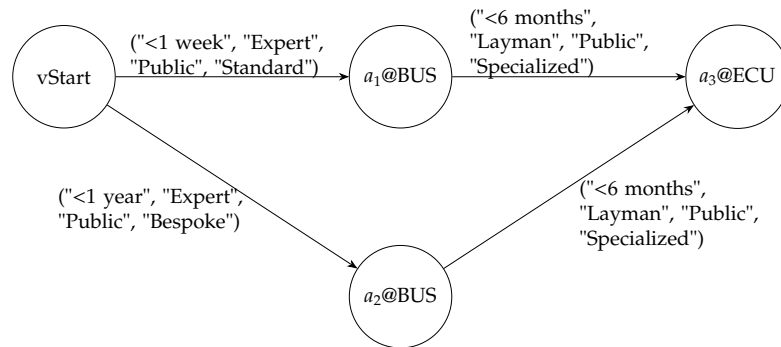


Figure 4.3.: Example of Attack Graph

### 4.3.6. Threat Path Extraction

In this step, we map predefined threats to the path nodes, which consist of an attack step and a location. A threat may have multiple targets, such as performing a DoS attack on a bus or shutting down an ECU. We extract the shortest path to each target of a threat from the precomputed shortest paths (see Algorithm 3). Starting from the virtual start node ( $vStart$ ), we calculate the shortest path to each target. We then compare the shortest paths to different targets of the same threat and select the shortest one. This results in the shortest path for each threat.

#### Example

In our example, the BellmanFord algorithm calculates all the shortest paths from  $vStart$ . In this example, the resulting paths are:  $vStart \rightarrow a_1$ ,  $vStart \rightarrow a_2$ , and  $vStart \rightarrow a_1 \rightarrow a_3$ .

---

---

**Algorithm 3** Extract shortest attack path of threat

---

```
1: Path  $\leftarrow \emptyset$ 
2: for all target  $\in$  threat do
3:    $T \leftarrow$  target
4:   Pathtarget  $\leftarrow$  target
5:   while  $T \neq vStart$  do
6:     print  $T$ 
7:     Pathtarget  $\leftarrow$  Pathtarget  $\cup$  shortestPaths.predecessor[ $T$ ]
8:      $T \leftarrow$  shortestPaths.predecessor[ $T$ ]
9:   end while
10:  if weight(Pathtarget) < weight(Path) then
11:    Path = Pathtarget
12:  end if
13: end for
14: return Path
```

---

#### 4.3.7. Most Feasible Path Calculation

In the final step, we calculate the feasibility of the most feasible attack path for each threat. We combine the maximum attack feasibility values from each rating category to determine the feasibility of the path.

##### Example

In our example with the three paths:  $vStart \rightarrow a_1$ ,  $vStart \rightarrow a_2$ , and  $vStart \rightarrow a_1 \rightarrow a_3$ , but we predefined only one target ( $a_3$ ). Our method extracts all the shortest paths from the graph that lead to this specific attack representing the threat. In this case, the shortest path to  $a_3$  is  $vStart \rightarrow a_1 \rightarrow a_3$ . The final attack path consists of two attack steps:  $vStart \rightarrow a_1$  (" $<1$  week", "Expert", "Public", "Standard") and  $a_1 \rightarrow a_3$  (" $<6$  months", "Layman", "Public", "Specialized"). To calculate the attack feasibility of the complete path we take the maximum required expertise ("Expert"), the maximum required knowledge ("Public"), and the maximum required equipment ("Specialized"). We sum up the required time where  $a_1$  has an estimated time of two weeks and  $a_3$  has an estimated time of four months resulting in the category " $<6$  months". Therefore, the resulting attack path is rated as (" $<6$  months", "Expert", "Public", "Specialized").

---

By following these automated steps, we can extract attack paths efficiently and determine the feasibility of the most probable attack paths for different threats in our vehicle model.

#### 4.4. Related Work on Threat and Risk Analysis

Several projects and standards have proposed schemas or processes for a TARA in the automotive domain. One of the earliest methodologies proposed was the EVITA project [177] in 2009, which introduced a security risk assessment methodology [156] for automotive E/E systems based on ISO/IEC 18045:2008 [87]. This approach has been adopted in many subsequent research projects, including HEAVENS [75] and SAE J3061 [159]. In 2014, the National Highway Traffic Safety Administration (NHTSA) [125] proposed a composite threat model for the automotive industry. In 2017, the European Telecommunications Standards Institute (ETSI) released several versions of a Threat, Vulnerability, Risk Analysis (TVRA) [51], which focuses on telecommunications threats. SAE J3061 [159], published in 2016, also mentions the risk assessment methods of EVITA [156], an early version of TVRA [51], and HEAVENS [111] as the basis for its task. In [161], an application of this SAE method to a communication controller is shown.

Next to these standardization activities, several research approaches suggest improvements. In 2015, the SAHARA [121] method was proposed, combining safety and security analysis into one strategy. SAHARA uses the STRIDE model to identify security hazards during a safety analysis. Macher et al. also reviewed recommendations for a TARA process in the SAE J3061 and different threat analysis methods in [120]. RACE [22], which combines EVITA with TVRA, was also proposed the same year, along with a risk calculation using the EVITA controllability concept. In 2016, Sommer et al. presented a classification of automotive attacks in [171], where the authors present a classification schema containing a vulnerability rating that uses the common vulnerability scoring system (CVSS). The work focuses on categorizing existing attacks contrary to our approach of generically generating attack paths. Monteuis et al. proposed the SARA framework [132], which improves a threat model and introduces a new attack observation metric focusing on driver assistance systems. In 2019, Bolvinou et al. suggested a controllability-aware framework named TARA+ [20] for automated driving vehicles fusing features of the previously introduced SAE and ISO standards, and Maple et al. published a reference architecture for attack surface analysis of smart cars in [123]. The reference architecture of Maple et al. focuses on the functions of vehicles rather than a detailed architecture with communication technologies and different ECUs, which results in a more abstract attack tree.



---

The introduction of the ISO/SAE 21434 [95] in 2021 replaces the SAE J3061 "Cybersecurity Guidebook for Cyber-Physical Vehicle Systems" [159]. One aspect of this standard is the attack surface assessment with a TARA. Dantas et al., in their work [42] exemplify how an incremental security engineering approach based on this standard can help engineers argue the assets' security claims.

Our work focuses on generically identifying the assets of modern electric vehicles that form the attack surface and executing a feasibility assessment for attacks on these assets. The concept of an attack tree introduced by Schneier [162], and unified parametrizable notation for attack trees, proposed by Wang et al. [191], forms the basis of our approach. Attack graphs, introduced by Phillips and Swiler [147], are closely related to attack trees but comprise all possible attack paths rather than just one goal. Rieke introduced the computation of compact representations of the graph and the inclusion of liveness analysis in [153]. In their work, Manadhata and Wing [122] provide a formal model of an attack surface.

Our approach is closest to the evaluation of the required attack potential identified using attack trees in EVITA Deliverable 2.3 [156] and the attack surface tables proposed by Petit and Shladover [145], with the exception that our work considers important assets of current vehicles that these tables could not cover. We demonstrate how our method can be used to evaluate specific attack scenarios, which can be integrated into the process described in ISO/SAE 21434. Our approach also allows for the evaluation of various mitigation strategies to secure vehicle networks and their impact on the overall attack feasibility. This contributes to the goal of designing a secure vehicle.

To extend the idea, the work [172] connects threat descriptions of UN/ECE 155 [184], Microsoft STRIDE, Common Attack Pattern Enumerations and Classifications (CAPEC), and Common Weakness Enumeration (CWE). It validates these using real-world attacks from the Automotive Attack Database (AAD), first introduced in [154].

## 4.5. Application to Reference Architecture

In this section, we apply our threat analysis method to the reference architecture introduced in Chapter 3. We start by converting the reference architecture into a graph representation, allowing us to identify potential entry points for the different attackers from our attacker model, as described in Section 3.3. Subsequently, we use this graph to generate potential attack paths based on the attack steps we identified and assessed in Section 4.2.2. By

---

identifying the shortest paths in the attack graph, we can pinpoint the most feasible ones in the reference architecture for various threat scenarios.

#### 4.5.1. Transferring Reference Architecture to Graph

We create nodes for each ECU and communication channel to represent the reference architecture as a graph. Each ECU is connected to the communication channels it interacts with, forming a directed graph. This approach offers the advantage of accommodating multiple ECUs connected to the same communication channel.

Based on the reference architecture presented in Chapter 3, we construct the graph depicted in Figure 4.4. The graph illustrates the relationships between ECUs and communication channels.

#### 4.5.2. Identifying Entry Points

Depending on the attacker model, potential entry points for attackers exist. Considering the three attacker types from our model (*Local Attacker*, *Advanced Remote Attacker*, and *Internal Attacker*), distinct entry points into a vehicle's network can be identified:

*Local Attacker*: This attacker could introduce a device into the vehicle or connect to the OBD port. This would grant the attacker access to various bus systems without direct access to cryptographic materials for tasks like crafting SecOC messages.

*Advanced Remote Attacker*: This attacker would have access to an ECU and could execute actions associated with that ECU's capabilities. Potential target ECUs could be a TCU or an ECU in the infotainment domain.

*Internal Attacker*: This attacker would have access to the charging infrastructure (CS or CSO), which operators with limited security expertise might manage.

#### 4.5.3. Generating Attack Paths

We generate potential attack paths using the graph representation based on the steps identified in Section 4.2.2. We assigned a location and the necessary attacks that needed to be executed before the attack step for every attack. E.g., for the attack 3.23 On-car Communications (legit): CAN/CAN FD, FlexRay, Ethernet, the attacker

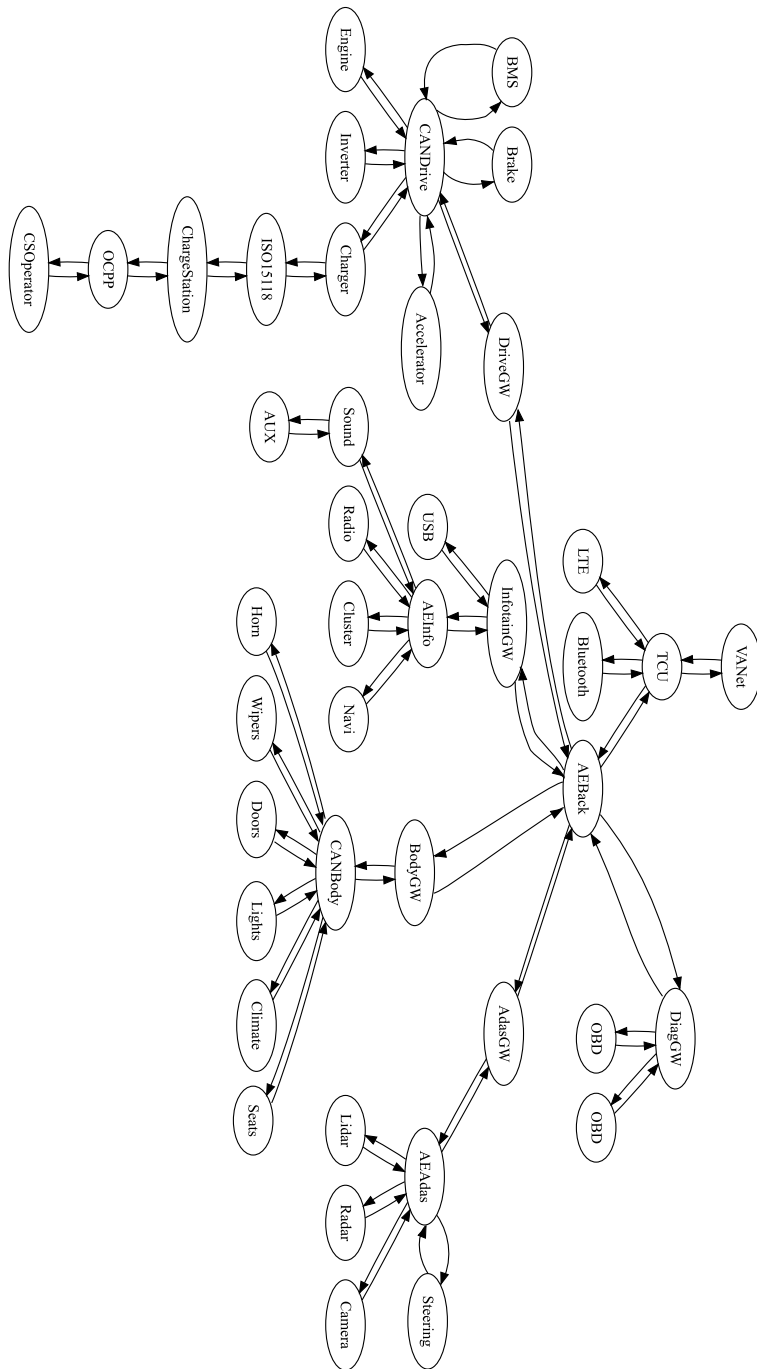


Figure 4.4.: Generated Network Graph from Reference Architecture

---

needs to have control over a regular ECU (Attack 4.17). Possible locations for Attack 3.23 are any CAN or Automotive Ethernet networks. We modeled a potential attack graph with all attack steps where every node is an attack step at a specific location in our reference architecture. Two nodes are connected if these are at the same or an adjacent location (e.g., the TCU is adjacent to the Long Term Evolution (LTE) network), and the first node is a condition to execute the second node. The weight of each edge is defined with the attack feasibility of the second node.

Furthermore, we introduced a start node connected to all possible entry points of an attacker described in the previous section. Again, the weight of the edge is determined by the following attack step.

#### 4.5.4. Identifying Attacker Goals

For further preparation in the analysis to determine the most feasible attack paths in the TARA, it is crucial to identify the potential targets that attackers may pursue first. Given the multitude of ECUs and features within vehicles, attackers could have a wide range of attack goals. These might include unauthorized feature unlocking, odometer manipulation, vehicle theft, engine manipulation, emission control tampering, and more. However, to analyze the threat landscape effectively, it is essential to narrow the focus to a subset of high-impact attack goals.

In the context of the vehicle's safety, the most critical threat scenario involves attacks that can influence driving functionalities, potentially leading to catastrophic accidents involving the vehicle and posing risks to other road users. Within this safety-focused framework, two primary attack types are of significant concern:

*ECU Command Execution:* Attackers may attempt to execute unauthorized commands or code within an ECU, which can result in unintended vehicle behavior. This can involve actions like accelerating or braking the vehicle. Such attacks could target ECUs within the ADAS domain or the drive domain.

*On-Car Communications Manipulation:* Attackers might manipulate on-car communications, such as the CAN/CAN FD or Automotive Ethernet protocols, to inject malicious data or alter legitimate messages. These attacks could impact both the drive and ADAS domains, potentially leading to safety-compromising scenarios.

Table 4.7 enumerates the specific attacks and their corresponding potential impact locations, focusing on ensuring the safety of passengers and other road users.

Table 4.7.: Final Attack Step to Impact the Vehicle Safety

Attack	Location
3.19 On-car Communications (replay): CAN/CAN FD, FlexRay, Automotive Ethernet	CAN Drive ADAS AE
3.20 On-car Communications (inject): CAN/CAN FD, FlexRay, Automotive Ethernet	CAN Drive ADAS AE
3.21 On-car Communications (alter): CAN/CAN FD, FlexRay	CAN Drive
3.22 On-car Communications (alter): Automotive Ethernet	ADAS AE
3.23 On-car Communications (legit)	CAN Drive ADAS AE
4.17 On-car ECU (execute Code/Commands)	ADAS GW Drive GW Steering Accelerator Brake Engine Inverter

It is important to note that specific attacks, such as unauthorized door unlocking or engine starting for theft purposes, may leverage the same attack methods. Additionally, manipulations that do not directly impact vehicle safety, like engine tuning or odometer tampering, may require persistent changes to vehicle ECUs.

Beyond the vehicle, the emerging electric charging infrastructure introduces risks both to drivers and the power grid. This has been elaborated in detail in our previous work [212] and in [99] for example. Attackers targeting the charging process could aim to manipulate charging synchronization or exploit the payment system’s vulnerabilities. While the financial gains from such attacks might be limited due to the relatively low cost of charging (about 15 € based on<sup>1</sup>), the payment process also involves valuable personal data.

Attacking the power grid via the vehicle charging system requires accessing CSOs or CSs to execute commands or manipulate charging processes through protocols like OCPP or ISO 15118. Table 4.8 provides detailed possible final attack steps. For data theft, attackers might engage in MitM attacks to intercept communication between the vehicle

<sup>1</sup><https://alternative-fuels-observatory.ec.europa.eu/consumer-portal/electric-vehicle-recharging-prices>

and charging infrastructure partners. Table 4.9 shows the possible attack locations and final attack steps.

Table 4.8.: Final Attack Step to Influence the Power Grid

Attack	Location
3.13 Wired Com. (intercept, alter, inject, replay): PLC	PLC
2.16 Wireless Com. (intercept, alter, inject, replay): Cellular (LTE/5G)	OCPP
* Control External Device or Server	CS CSO

Table 4.9.: Final Attack Step to Gain Access to Private Information

Attack	Location
3.13 Wired Com. (intercept, alter, inject, replay): PLC	PLC
2.13 Wireless Com. (listen): Cellular (LTE/5G)	OCPP
* Control External Device or Server	CS CSO

Notably, while many relevant attacks extend beyond the vehicle network itself, they are interconnected with the extended communication network involving various backend systems. This highlights the complexity of safeguarding against these threats and underscores the importance of a holistic security approach.

Finally, the attack on sensors presented in [169] or [29] may introduce catastrophic outcomes in the future, especially concerning autonomous driving; however, in our model, these are still assisting systems for the driver with limited effects, to the driving behavior. These attacks will become more relevant in the future of automotive security and thus be part of future work.

#### 4.5.5. Identifying Feasible Attack Paths

With all the components, the reference architecture, the rated attack steps, the attack goals, and the entry points, we can now calculate the most feasible attack paths using our newly introduced algorithm. The final results presented here are split into attacks against the vehicle's safety and the charging communication, which would result in catastrophic impacts if attacked successfully.

---

#### 4.5.6. Feasible Attack Paths against Vehicle Safety

When analyzing possible attack paths for the *Local Attacker* targeting vehicle safety, two fundamental routes come up based on our algorithm and the specified entry points and targets.

In the first path, the attacker may gain control of an ECU to execute code. In this scenario, the attacker seeks to gain control over an ECU to execute unauthorized code. The attack path involves accessing the bus of the targeted ECU (Attack 3.1), bypassing or overcoming secure communication mechanisms (Attack 1.8), communicating with the ECU (Attack 3.19), executing an exploit (Attack 4.3), and ultimately taking control over the ECU (Attack 4.17). According to our hybrid assessment method, this attack path has a low feasibility rating, with an elapsed time of less than 6 months, proficient specialist expertise, restricted knowledge of the item, moderate opportunity, and standard equipment. According to ISO 21434, this results in a very low attack feasibility.

The second possible path for an attacker is to manipulate the internal vehicle communications: In this case, the attacker aims to manipulate on-car communications, such as CAN/CAN FD, FlexRay, or Automotive Ethernet, by replaying a message (Attack 3.19). This attack path is characterized by high feasibility, with an estimated elapsed time of less than 1 week, proficient specialist expertise, restricted knowledge of the item, moderate opportunity, and standard equipment.

Different feasible attack paths emerge for the *Advanced Remote Attacker*, who is limited to the infotainment domain or the TCU. The most practical path involves taking over a safety-critical ECU, starting with control over the TCU (directly connected to the Automotive Ethernet backbone). From there, the attacker can execute commands on the TCU (Attack 4.17), sending legitimate commands (Attack 3.23) to the gateway of the drive domain. These legitimate commands can trigger legal communication forwarded to a safety-critical ECU, which needs to be exploited (Attack 4.3) to execute commands locally (Attack 4.17). This attack path is assessed as having low feasibility, with an elapsed time of less than 6 months, proficient specialist expertise, restricted knowledge of the item, an unlimited window of opportunity, and standard equipment. Injecting messages into a safety-critical bus system is considered less complex, with a high feasibility rating based on the same parameters.

In summary, our reference architecture is highly vulnerable to attacks on in-vehicle communication. For both attacker types in our model, the most straightforward attack path, as determined by the presented algorithm, has a high feasibility rating. As a result,

---

---

we will investigate the security of in-vehicle networks in the following chapters, specifically focusing on the security of CAN and Automotive Ethernet.

#### 4.5.7. Feasible Attack Paths against Charging Communication

Regarding attacks on charging communication, the first attacker model (*Local Attacker*) is focused on attacks against the PLC and ISO 15118 communication. Feasible attack paths involve physical tampering with the PLC (Attack 3.6) and altering or injecting PLC messages (Attack 3.13), both of which result in a medium attack feasibility rating. Furthermore, the attacker must bypass the security of TLS, which, if properly implemented, results in a very low attack feasibility rating.

Similarly, extracting personal data relies on the proper use of TLS, leading to attack paths that are similar in feasibility.

For further communication involving CS to CSO, CSO to CCH, and CCH to MO, the attack path only slightly differs in the low-level attack. Attacks against cellular communication involve accessing cellular communication (Attack 2.2) instead of physical tampering (Attack 3.6) and using cellular communication to alter or inject messages (Attack 2.16) instead of PLC (Attack 3.13). These paths also result in a medium or very low attack feasibility if TLS is implemented correctly.

The *Advanced Remote Attacker* can directly control charging limits in the infotainment ECU without additional steps. However, attacking the charging communication or backend components results in very low attack feasibility. In our reference architecture, the attacker must first escape the infotainment domain, enter the drive domain, and take over the charger ECU before being able to perform attacks against the CS.

The final attacker in our model, with access to a backend system, has a straightforward path to gather information about a charge. This is because all information is exchanged in clear text among communication partners, including the vehicle, CS, CSO, CCH, and MO. Additionally, entities like CS and CSO can control the charging process, influencing energy consumption in the grid.



---

## 4.6. Summary of Risk Assessment for Modern Electric Vehicle Reference Architecture

A TARA plays a pivotal role in the cybersecurity engineering of modern electric vehicles. Central to risk determination is the thorough assessment of the attack surface, accompanied by the feasibility evaluation of potential attacks on each asset within a vehicle. In this chapter, we introduced a novel approach for assessing the attack surface, involving identifying assets, threat scenarios, analysis of attack paths, and evaluating attack feasibility, all in alignment with ISO/SAE 21434 standards. We also presented an automated method to expedite this process, reducing errors and enhancing efficiency. Our attack feasibility assessment serves as a valuable tool within a TARA, offering a holistic view of potential threat scenarios and their likelihood. The automation of this process provides a quicker and less error-prone alternative to manual attack derivation. Moreover, it demonstrates how introducing security mechanisms can reduce attack feasibility, aiding in evaluating various security technologies for safer vehicle architectures.

We explored three high-impact risks: manipulating driving behavior, privacy breaches in the charging process, and threats to power grid stability through electric vehicle charging. The application of our algorithm to these threat scenarios and our three attacker models, considering our reference architecture, revealed paths with high or medium attack feasibility. Building on these observations, our work in the next chapter delves into the security of CAN communication, focusing on analyzing these networks against potential vulnerabilities and securing them. We also investigate the protection of Automotive Ethernet with a special emphasis on scenarios where a single ECU may be compromised in Chapter 6. Lastly, we will check the security of electric charging communication, particularly assessing the reliance on TLS for all connections and conducting a privacy analysis to address the distribution of personal information among charging process entities. By addressing these crucial aspects of vehicle cybersecurity, we aim to contribute to developing safer and more secure electric vehicles in the rapidly evolving automotive landscape.



---

## 5. Controller Area Network (CAN) Bus Security

---

Modern vehicles rely heavily on the CAN bus and its predecessors - CAN FD and CAN XL- to control most of their safety-critical and non-critical functionalities. However, these bus systems are not immune to attacks, where malicious messages can be injected or manipulated. In this chapter, we analyze the security of various countermeasures for CAN bus attacks and present our solution for a secure bus communication technology.

Most countermeasures use Message Authentication Code (MAC) coupled with time stamps or message counters to provide message freshness. The most prominent example of such a countermeasure is SecOC, which is standardized in AUTOSAR. The TARA we have performed in Chapter 4 suggests a potential risk for CAN communication, especially for a *Local Attacker*. For this reason, we analyze different countermeasures and derive a generic model for secure CAN communication based on the security evaluation in this chapter. Furthermore, we formally verify the security properties of this generic model. The analysis showed some limitations with the generic model regarding the freshness of messages, which were addressed in a self-developed solution called BusCount. This solution is practically evaluated, and a formal proof verifies the security properties of the solution. The two approaches, BusCount, and the generic model are compared regarding their security properties in this chapter.

This chapter is based on the results of two papers [67] and [202]. The original manual proof of both papers was the responsibility of Sigrid Gürgens, while I designed the protocol and implemented the solution on an FPGA. For this thesis, I conducted an automated proof using Tamarin, which yielded the same results as the proofs in the cited publications.

To begin with, the chapter outlines the protection goals for CAN bus communication and describes the state-of-the-art protection mechanisms in CAN. The following sections describe the generic model followed by the self-developed solution, and finally, their effectiveness against a *Local Attacker* or *Advanced Remote Attacker* is evaluated.

## 5.1. Setting

Chapter 3 describes the general reference architecture, also used in this part of the thesis. Here, the focus is on the domains that still use CAN or CAN FD. Figure 5.1 shows the energy and drive domain where the domain controller is connected with the brakes, battery management, engine, accelerator, inverter, and charger in a CAN FD bus. Its basic communication technique is described in Section 2.1. An attacker in these networks is either a *Local Attacker* or an *Advanced Remote Attacker*, as introduced in Section 3.3, since the *Inside Attacker* is only valid in communication scenarios with external entities. First, a definition is given of the protection goals needed to be fulfilled to mitigate attacks by the defined attackers. An overview of existing protection strategies is given in the second part of the section. This overview is used in the following section to derive a generic model of existing techniques.

This overview is used in the following section to derive a generic model of existing techniques.

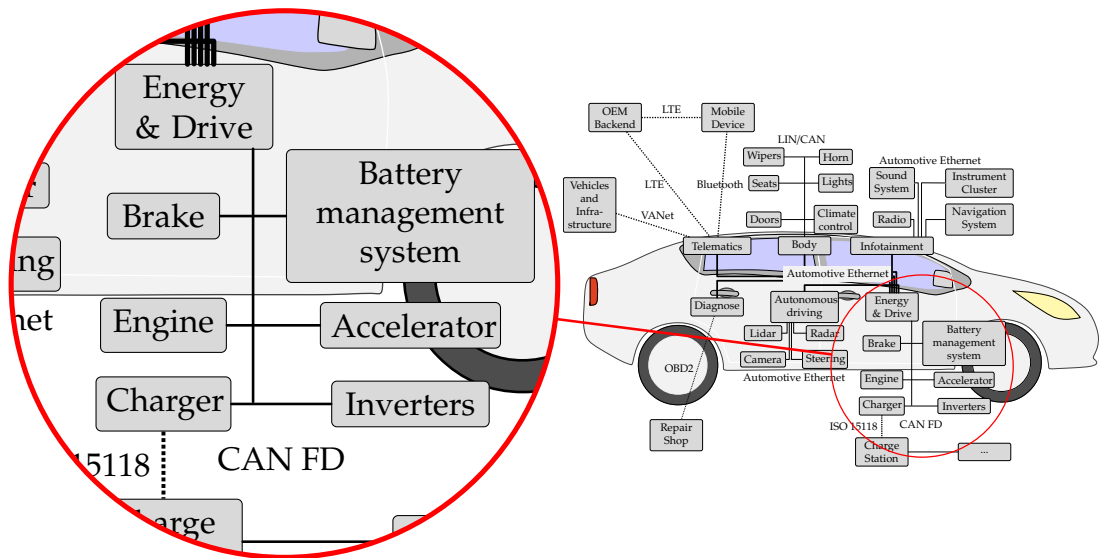


Figure 5.1.: Example of a CAN Network

---

### 5.1.1. Protection Goals

To prevent vulnerabilities introduced in the past, a secure CAN bus communication in a vehicle must adhere to a set of specifications, such as data availability and integrity. These are the *Data Origin Authenticity*, *Immediacy*, and *Non-repeatability*. The data origin authenticity ensures that the data received are from an authentic entity and have not been altered. Furthermore, the *immediacy* of transmissions is essential. It ensures the timeliness of a message, which is especially significant for controlling cyber-physical systems. Last, *non-repeatability* ensures that a receiver will not accept a valid message twice, preventing replay attacks. Compared to most related work, a distinct separation between the two properties, *non-repeatability* and *immediacy*, is made. Many works only have a general security goal called *message freshness* to protect against message replay. However, it does not cover whether a message is delayed in any way. In the following, these specific requirements are defined, which are necessary for protecting the CAN bus communication and, thus, the safety of the passengers of a vehicle.

- $S_1$  *Data Origin Authenticity*: Transmitted messages in an in-vehicle network must only be accepted if and only if they were sent by a legitimate network member and were intended for the recipient. This characteristic makes it impossible for attackers to alter messages or send them through replaced components or third-party devices on the bus system without the notice of the receiver. A more robust form of this property is to demand that a message must be accepted if and only if it was sent by the rightful entity in the network. This property excludes any attackers that take control over an ECU that is not directly part of the communication.
- $S_2$  *Immediacy*: In contrast to classic communications in a computer network, processing signals in in-vehicle networks within a tight time period is critical. Exceeding the predefined time window can result in catastrophic accidents, even if the message is authentic, as described in the previous property. One example would be a delay of signals of the anti-lock braking system, which can cause the braking of tires to occur at the wrong time. The property *immediacy* is characterized by the trait that a message sent at  $t_1$  is only processed until  $t_2$  if and only if the difference between  $t_2$  and  $t_1$  is smaller than a predefined time limit.
- $S_3$  *Non-repeatability*: Next to *data origin authenticity* and *immediacy*, *non-repeatability* is the third property needed for a CAN network. This property says that if a message in a network is accepted at time  $t_1$ , then the same message must not be accepted again later. With this characteristic, attackers are not able to inject messages that were previously recorded.

---

### 5.1.2. State of the Art CAN Bus Security

Many secure communication technologies have been discussed in literature, and standards have been introduced for the unique limitations of CAN bus communication. A very early solution was presented in [164] with a truncated MAC, which is a basis for many of these solutions. In the following, an outline of the various technologies is presented that can be clustered into the following categories: Central component approaches, using a central component to identify intrusive messages, and decentralized systems.

Most security approaches for CAN buses are based on sending a MAC of the transmitted data to authenticate the message. However, there are still approaches without replay protection since no freshness value is included in messages. One example is the approach by Bella et al. [13]. They introduce the TOUCAN protocol that suggests appending a Chaskey MAC to the payload and encrypting the entire message using SPECK64. The size of the MAC is reduced to 24 bits. Next to this approach, Hazem et al. presented LCAP [74], which uses an unconventional approach to protect against replay attacks compared to most security solutions for CAN buses. Each CAN message secured with LCAP contains a truncated part of a hash chain. The message is then encrypted. Woo et al. [198] suggest a regular renewal of the MAC keys to counter attackers that replay messages. Recently, Groza et al. [65] presented an approach to cycle through CAN IDs based on a customized cryptographic hash that maintains the message hierarchy. In predefined time intervals, the counter included in the MAC is incremented; thus, the IDs change. This approach increases the resistance against reverse engineering and denial of service attacks related to a specific ID. As mentioned in the paper, it does not provide data integrity and authenticity, which needs an additional security protocol. Moreover, freshness is not guaranteed since the counter used in the MAC of the CAN ID does not change with every message. In the case of constantly changing counter values (IDs) and if a significant limitation of ID range is acceptable, this can be a viable alternative to transferring fresh counter values.

Nürnberg and Rossow [138] developed VatiCAN, an HMAC-based authentication mechanism that transmits a MAC in a second separate message following the original CAN message. With a delay of about 4 ms, the receiver validates the MAC. Replay protection is realized with a monotonically incremented counter, its starting value being a random nonce generated by a central component for every message ID. The authors recommend performing this procedure only for a limited number of critical messages since it increases the bus load. Van Bulck et al. improved this approach in [25] by introducing software isolation, attestation, and key update mechanisms.

Hartkopp et al. presented a further approach to introduce freshness to CAN messages.

---

MaCAN [71], formally verified in [24], introduces a central trusted time server that distributes time information over the network. This information is used as a freshness value for message authentication.

AUTOSAR specifies the protection of communication in in-vehicle networks based on a MAC and a freshness value. The specification of the Secure Onboard Communication (SecOC) [8] module suggests adding a truncated timestamp or message counter and a truncated authenticator to every message. A specific counter mechanism is based on splitting the counter (with a maximal length of 96 bits) into three different parts, named *trip counter*, *reset counter*, and *message counter*. As the name suggests, the *trip counter* changes every new trip. The *reset counter* resets periodically, and the *message counter* is incremented with every message. Only the trip counter is stored persistently in the device storage to prevent the loss of counter values in case of power loss of an ECU. A message contains a truncated freshness value with a length between 0 and 8 bits. The truncated authenticator consists of the first 24 to 28 bits of the MAC covering the full freshness value and the message.

Similar to SecOC, many approaches in literature use counters and an application-level protocol to ensure replay protection. Kurachi et al. [109] suggest attaching a truncated MAC (8-bit) and a truncated monotonic counter (4-bit) to a message. A monitoring node verifies messages during transmission and overwrites invalid messages with an error frame. ECUs do not verify messages. Groll et al. [64] suggest an initialization phase to form groups of ECUs. These groups generate a shared symmetric key using an asymmetric key exchange. ECUs use the shared secret within these groups for authentic and confidential encryption. A counter should be part of the message to protect against replay attacks. Lin et al. presented an approach in [118] with symmetric keys for message authentication. A sender calculates a MAC for every receiver. Every ECU also holds two counters for replay protection per message ID, the last counter it has sent and the last one it has received. Every receiver can verify the MAC and process its corresponding message. The LeiA protocol by Radu et al. [150] is another solution that transfers MAC and counter value in a separate message. Every ECU has a session key for each relevant message ID derived from a long-term symmetric key and renewed after a certain period. VeCure [192] is a CAN authentication framework similar to VatiCAN. The authentication value is also transmitted via a separate message, but contrary to VatiCAN, the second message includes a *Node-ID* besides a *Message Counter* and a four-byte HMAC value.

Alternatively, several approaches suggest using CAN+ [213], a protocol extension for CAN, allowing the transport of 120 bits of additional data. The first approach is CANAuth, presented by Van Herrewewege et al. [187]. Another one is LiBrA-CAN [66]. LiBrA-CAN

---

introduces (Linearly) Mixed MAC, which mixes multiple MACs of one message generated with different keys, allowing receivers to verify a MAC even though they do not know all keys. The approach ensures that receivers cannot impersonate a sender in an adequately organized group. Both approaches send counter values in their messages to protect against replay attacks.

Some works are also considering the implementation of a secure CAN bus controller. Their approaches introduce the calculation of MACs, denial of service countermeasures, or intrusion prevention mechanisms. [167] implemented a CAN controller that included key generation and storage as well as a physical unclonable function, as well as encryption and decryption, allowing authenticated communication over CAN, but it does not provide any type of replay protection. Similarly, Ueda et al. presented a CAN controller with integrated HMAC in [183]. To ensure replay protection, a truncated monotonic value of 4 bits is part of every message. Messages that are not authentic are destroyed while correct messages update the counter.

We observed that most of the presented approaches (cf. Table 5.1) have similar ways to ensure replay protection and authentication of messages. All approaches add a MAC to a CAN message. A sufficient replay and delay protection requires a combination of a MAC, which provides the authenticity of a message with a freshness value. Most approaches introduce a counter value to provide freshness since the usage of time or nonce values has disadvantages, as discussed for example in [214]. The transmission of MAC and freshness values is either realized in an additional message or achieved by including truncated values in the same message. The receiver performs the verification of a complete message or an additional node. In the following section, we present a detailed generic model covering the characteristics of the current counter-based approaches for freshness. This model is then compared to our approach based on formal verification of the security goals.

## 5.2. The Generic Counter Concept

Regarding the recent research on secure CAN communication, which we discussed in Section 5.1.2, several solutions show similarities. These are simplified to a generic model for the security evaluation of high-layer security protocols in a CAN bus network. In this section, we introduce this generic model, which uses a counter-based freshness value and an arbitrary MAC like a majority of presented solutions in industry and academia.

The generic protocol flow is depicted in Figure 5.2, which shows three ECUs (ECUs *A*, *B*, and *C*) connected via CAN. The figure illustrates the sending and receiving of messages



Table 5.1.: Comparison of different authentication approaches for CAN-Bus (C: Counter, T: Timestamp, N: Nonce, AID : Authenticated ID, H: Hash Chain, K: Key Renewal, \*: not described)

	HW change	SW change	Central component	Freshness technique	MAC	Encryption	Transfer techniques for MAC	Synchronization of freshness value
AUTOSAR [8]	-	✓	-	C/T	✓	-	28 bit data field	✓
CaCAN [109]	✓	✓	✓	C	✓	-	8 bit data field	-
CANAuth [187]	✓	✓	-	C	✓	-	CAN+	(✓)
Groll et al. [64]	-	✓	✓	C	✓	✓	*	-
LeiA [150]	-	✓	-	C	✓	-	separate message	✓
LibrA-CAN [66]	-	✓	✓	C	✓	-	CAN+	-
Lin et al. [118]	-	✓	-	C	✓	-	*	(✓)
Ueda et al. [183]	✓	✓	✓	C	✓	-	8 bit data field	-
VeCure [192]	-	✓	-	C	✓	-	separate message	-
MaCAN [71]	-	✓	✓	T	✓	-	32 bit data field	✓
VatiCAN [138]	-	✓	✓	C+N	✓	-	separate message	✓
vulCAN [25]	✓	✓	✓	C+N	✓	-	separate message	(✓)
Woo et al. [198]	-	(✓)	(✓)	K	✓	✓	CAN-FD	-
LCAP [74]	-	✓	-	H	-	✓	16 bit extended ID	✓
TouCAN [13]	-	✓	-	-	✓	✓	24 bit data field	-
Siddiqui et al. [167]	✓	✓	✓	-	-	✓	data field	-
CAN-TORO [65]	✓	✓	✓	AID	-	-	-	(✓)

---

where ECU  $A$  sends a message that ECU  $B$  and  $C$  receive. Every ECU starts in an idle state until  $A$  starts the sending process of a CAN message  $m$ , which is composed of the CAN ID and the message payload  $msg$ . ECU then prepares the sending process by incrementing its local counter and calculating the MAC for the message. The tag  $t$  is calculated with the MAC function over the message  $m$ , the local counter  $c_a$  using a cryptographic key  $k$ . In the final step, ECU  $A$  concatenates  $m$ ,  $c_a$ , and  $t$  and transmits this message. Note that most protocols use some sort of truncation to reduce the payload of the message or an additional transmission is used to transmit all values. After the message is sent, it is received by every ECU on the bus. ECU  $B$  and  $C$  now verify the message with two checks. First, the counter  $c_a$  in the message must be larger than the local counter  $c_b$  and the MAC tag  $t$  must be equal to the MAC calculation that ECU  $B$  performs with  $m$ ,  $c_a$ , and the key  $k$ . If one of the checks fails, the message is discarded, and the ECU waits for the following message. If both checks are valid, the message will be processed.

Most approaches do not explicitly introduce a synchronization mechanism for possible error (e.g., power loss or software failure) in an ECU that causes a counter loss. Often, a central component is used for the synchronization. One example is the SecOC standard [8], which suggests a freshness master. If a sender has an incorrect counter, such a dedicated entity or client must provide the correct counter value. In both cases, the counter value is transmitted in the payload CAN message with a reserved ID secured identically to regular messages.

Even though the generic counter protocol is relatively simple, it represents the characteristic properties of all the protocols mentioned above that increment counters after successfully validating the message. An essential characteristic property is that the local counters of message recipients only change when a message is accepted. Consequently, these protocols cannot prohibit so-called delay attacks, as will be formally shown in the next section. For such an attack, the adversary with the abilities described in Section 3.3 reads and then invalidates a message and all subsequent ones related to the same counter. The intended recipients will then accept the reinserted message at any later point in time if no further countermeasures are taken.

### 5.2.1. Formal Security Analysis of Generic Counter Concept

Based on the given generic protocol, we designed a formal model with the well-established Tamarin prover [12, 178]. We suggested using this in the work [110] for automotive protocols. The complete formal model consists of states and state transitions, which are

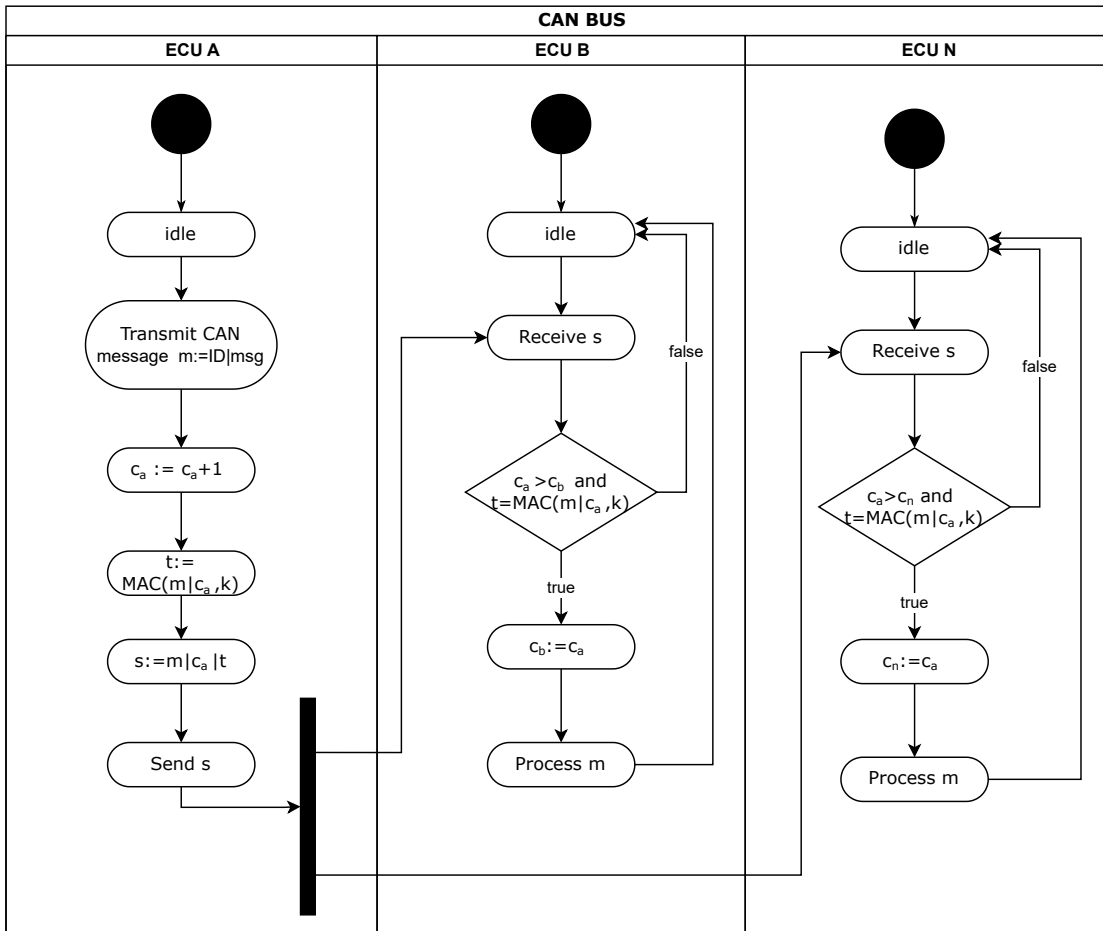


Figure 5.2.: Process of Generic Counter Communication

used to check security properties defined as lemmas. The complete formal model is shown and explained in Section A.1.1.

Our formal model, within the Tamarin prover, successfully demonstrates the properties of data origin authenticity and non-repeatability. However, the prover also uncovers a potential attack that violates the property of immediacy. This finding is crucial, as it reveals that an attacker could invalidate a message and repeat it in a different time phase. Consequently, the receiver cannot be certain that the received message was sent within a defined time frame and may be delayed by the attacker.

---

## 5.3. BusCount: A Low Layer Bus Counter Solution

Addressing the security flaw outlined in the previous section, we present BusCount. Our innovative approach to low-layer secure CAN bus communication guarantees key security properties such as immediacy, non-repeatability, and authenticity of messages (cf. Section 6.3.3). In addition to secure message transmission, our concept incorporates a mechanism for synchronizing freshness values, further enhancing the security of the system.

The core idea of BusCount is to make use of the fact that all participants of a bus system are able to read all transferred messages in this system. For this reason, the number of messages sent in the bus system is a value known by all attacked devices. This value can serve as a source of freshness for cryptographic operations without transmitting it. Furthermore, it changes with every transmitted message. This idea is applied to any bus network with the same properties.

We explain the protocol flow of BusCount in the following. Figure 5.3 illustrates this protocol.

The first step of transmitting message  $m$  in BusCount starts with decrementing the local counter (Note: Decrementation is only used for technical reasons explained in Section 5.3.1). Then, the MAC tag  $t_a$  is calculated over  $m$  and the local counter  $c_a$  with the cryptographic key  $k$ . In parallel the controller starts transmitting  $m$ . When  $t_a$  is ready and  $m$  is sent,  $t_a$  is added to the transmission. As soon as other ECUs start receiving a message, they decrement their local counters. Once the receiving of  $m$  is completed, the ECU calculates the MAC over its local counter  $c_b$  and the message  $m$ . Each receiver then validates whether the calculated tag equals the transferred one. Note that this means the counter of sender and receiver need to be equal. If the MAC is correct, the message can be processed. Otherwise, a receiver will overwrite the rest of the CAN message with an error frame. In contrast to the generic model, this protocol cannot synchronize counters and does not consume valuable space for a message. For this reason, we present a dedicated synchronization mechanism in the next section.

### 5.3.1. Synchronization

Counters may differ due to various reasons (e.g., an ECU is powered off or due to software faults). To ensure that this does not lead to a blocked CAN bus, every ECU can initialize a synchronization at any point in time. This synchronization allows all active participants

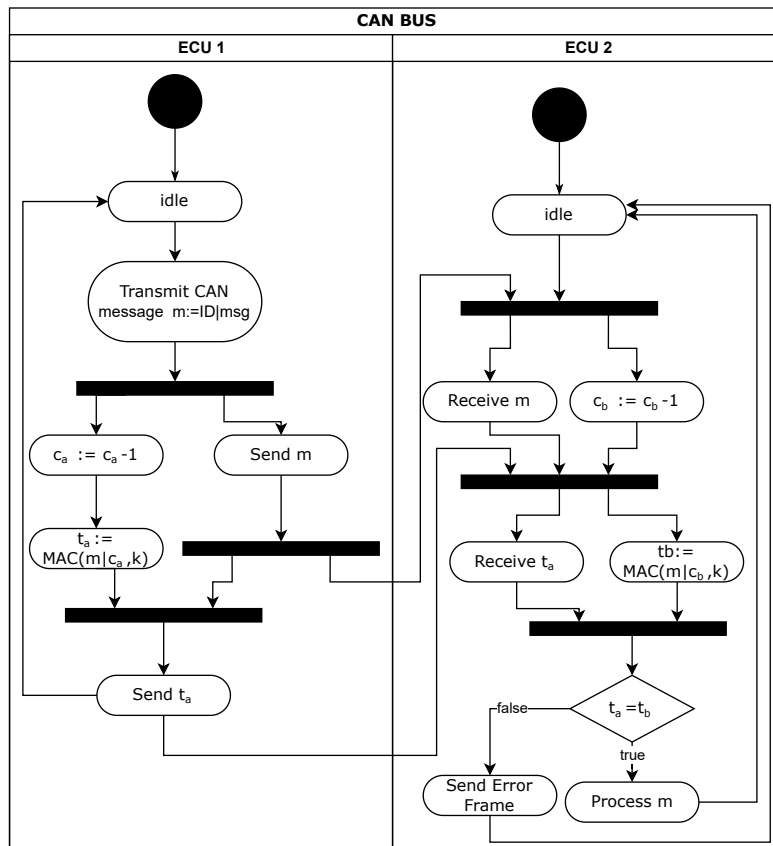


Figure 5.3.: Process of CAN message transmission of BusCount

in the bus to synchronize their counters within the transfer of one message by using the physical properties of the CAN bus. In particular, every 0 overwrites 1 sent via the bus simultaneously.

The synchronization method is exemplified in Figure 5.4. First, an ECU decides to start synchronization by sending a predefined ID. We suggest using ID=0 since this is the most critical message transferred since no other message can be transmitted until the counters are synchronized. Every ECU on the bus decrements its local counter, which is similar to a regular message. Now, every ECU transmits its local counter until the complete counter is sent or is overwritten at any bit. This mechanism is identical to the collision resolution of CAN (c.f. Section 2.1). The smallest counter will finally be transmitted

---

since a 0 is dominant over a 1. This explains why we decided to start a counter at the maximum value and decrement it in the protocol. The presented concept is also used in [134] to implement a key exchange over CAN. In parallel to the counter transfer, every ECU calculates the MAC over the lowest counter, and the ECU, which transmits its full counter, also transmits the MAC. Each ECU compares the transmitted value with the locally calculated MAC.

If both MAC values match, the ECU will update its local counter. Otherwise, an error frame invalidates the synchronization process.

### **5.3.2. Formal Verification of BusCount**

Similar to the generic counter model, we also verified the current BusCount protocol using Tamarin. In Section A.1.2, we describe the model of the BusCount message protocol in three steps: initialization, sending a message, and receiving a message.

Based on the formal model, all security properties can be proven to be correct. However, attackers can still perform DoS attacks. The results and the comparison between a generic counter solution and BusCount are discussed in the security evaluation in the next chapter, providing a comprehensive view of the protocol's strengths and weaknesses.

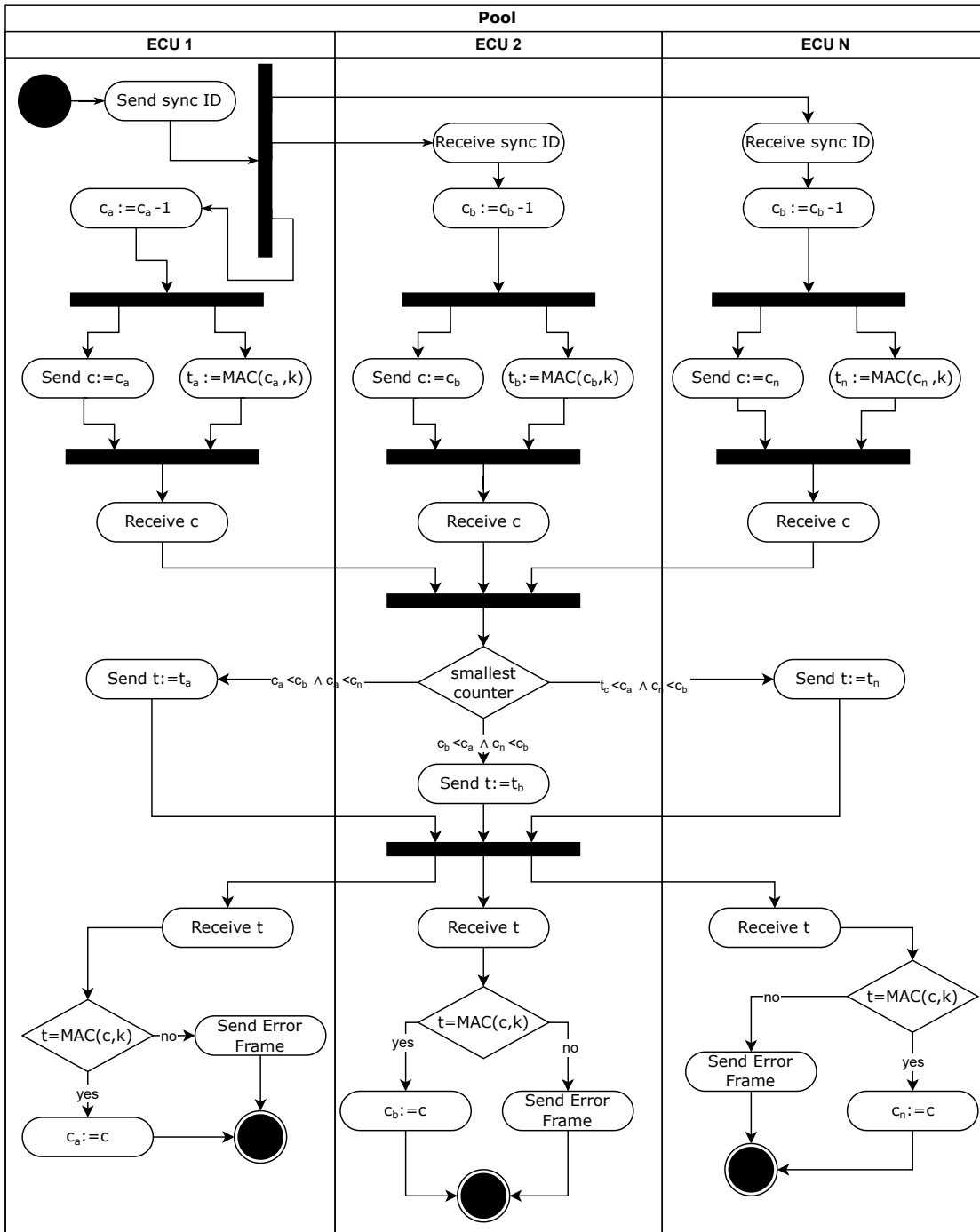


Figure 5.4.: Synchronization of BusCount

---

## 5.4. Evaluation

The evaluation chapter is divided into two sections. The first section discusses the security properties of BusCount and the generic model based on the formal analysis results. Using these results, we compare the two approaches. In the second section, we present a practical evaluation of our BusCount concept with respect to the limitations and requirements of CAN bus environments.

### 5.4.1. Security Evaluation

Based on the formal verification described in Section 5.2.1 and Section 5.3.2, we compare the security properties of both approaches. The central difference between these approaches is that the generic counter only changes its value when a message is successfully transmitted. In contrast, our BusCount decreases a global counter with every start of a message transmission. This integral mechanism ensures constantly changing freshness values. Since these freshness values are used to calculate the MAC of every message, an attacker can't replay or delay a message, as proven in the formal verification.

At the low layer of BusCount, all ECUs connected to the same bus can also validate the MAC of a message and send an error frame if the validation fails, preventing the transmission of the message. This further enhances the system's resistance even if some ECUs have outdated counter values.

The combination of these mechanisms allows for the formal verification of both immediacy ( $S_2$ ) and non-repeatability ( $S_3$ ) of our BusCount system. Additionally, the MAC ensures the authenticity of a message ( $S_1$ ). It is important to note that the properties of immediacy and non-repeatability only hold under the assumptions of our proof that at least one ECU has a correct counter. Therefore, it is necessary for the system to ensure that at the end of a trip when ECUs are shut down, at least one ECU stores the correct counter value in persistent storage. Furthermore, the system only works if all ECUs are constantly connected. In the event that an ECU becomes disconnected, an attacker is able to inject messages from other ECUs at a later point in time, as the disconnected ECU is not aware of any counter updates. Finally, we also require that there is no overflow in the counter value.

The generic counter model only provides the MAC to ensure authenticity, but its counter mechanism can be bypassed by altering the CRC of a message, for example. A local or



---

---

advanced attacker with the abilities described in Section 3.3 is capable of exactly this attack, which allows to violate the property of immediacy.

From a security perspective, our BusCount has the advantage over the generic counter model as it fulfills the property of immediacy. Furthermore, the update mechanism of BusCount allows the system to quickly recover from an incorrect counter of all ECUs. In contrast, traditional counter approaches rely on a central entity, such as a freshness master [8], which needs to track all counter values to some extent and becomes a single point of failure in the system.

In the following section, we discuss the size of the counter value and MAC that is sufficient to protect CAN messages, along with other implementation details of our practical evaluation.

#### 5.4.2. Practical Aspects

To conduct a practical evaluation of BusCount under realistic conditions, we implemented the protocol and performed tests in an automotive testbed. The results of these tests demonstrate both the feasibility of our approach and its potential for implementation in a cost-limited environment.

This section provides a detailed description of our setup, outlining the various components employed for the implementation and simulation of the vehicle network. We place particular emphasis on the design decisions we made, as these decisions significantly influenced the performance results obtained from our implementation.

##### Setup

The core part of our setup is an ECU with a CAN controller that implements our BusCount protocol. For this controller, we choose a low-cost FPGA in the form of an ICE40HX8K-B-EVN that has 7680 logic cells and runs at a frequency of 12 MHz. A CAN transceiver is necessary to convert signals from the FPGA to those used on the CAN bus to communicate with a bus. In our setup, this function is done by an MCP2561. The CAN controller is implemented based on an open-source CAN implementation<sup>1</sup> with the expansion of our protocol. Furthermore, we implemented an SLCAN<sup>2</sup> interface so that a connected ECU

---

<sup>1</sup><https://github.com/keesj/can-hdl>

<sup>2</sup><https://github.com/torvalds/linux/blob/master/drivers/net/can/slcan.c>

---

could send and receive messages. This ECU is emulated with an ARM1176JZF-S board running Linux with its default CAN driver SocketCAN<sup>3</sup>. We modified the driver to reduce the maximal payload of a message so the MAC has enough space in every message. For the evaluation, we have two identical setups that allow us to test send and receive as well as synchronization methods of BusCount. We also attached an automotive remaining bus simulator to verify if our setup is interoperable with the regular CAN protocol. This setup comprises a Vector VN5610 device and the PC simulation software CANoe v9. The entire hardware setup is displayed in Figure 5.5 and contains:

- **CAN controller:** ICE40HX8K-B-EVN
- **CAN transceiver:** MCP2561
- **ECU:** Raspberry Pi 2B
- **Remaining bus simulation:** Vector VN5610

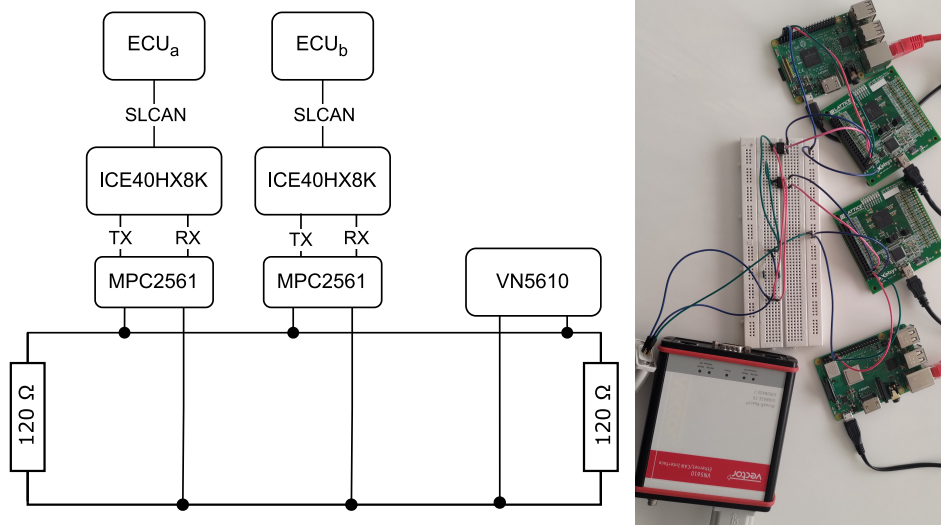


Figure 5.5.: Evaluation Setup for BusCount

<sup>3</sup><https://www.kernel.org/doc/Documentation/networking/can.txt>

---

## Design Decisions

In the development process, we made some decisions regarding the implementation of BusCount, which we explain in this section. First, we decided on the length of the transmitted MAC value; second, we decided on the size of the transmitted counter value; and last, we needed a reasonable counter size that would prevent replay attacks.

In compliance with AUTOSAR SecOC [8], which uses a 24-bit truncated MAC, we choose the same value for our approach. Due to the truncation of the MAC, it becomes more likely for an attacker to forge a correct message just by guessing. One of the most fatal attacks for the protocol would be to forge a synchronization message. For this reason, we integrate a counter of failed synchronizations similar to the error counter of CAN (c.f. Section 2.1). In the case of 16 failed synchronizations in a row or 128 failures in total, an ECU considers the bus untrustworthy. This means that the ECU should enable limp mode so that a driver can safely stop the car. An attacker thus has 128 attempts to forge a synchronization message successfully. Based on the size of the MAC, the chance of success for an attack is 0.000977% ( $\frac{\binom{128}{1} \binom{2^{24}}{127}}{\binom{2^{24}}{128}}$ ). A recovery mechanism is not part of the protocol.

Next to the MAC size, we also defined the size of the transferred counter. Due to the size of the MAC, only 40 bits of the CAN message are left. Since this is already a very limited payload, we decided not to transmit the counter value at all. Due to our protocol's high-speed synchronization mechanism, we can assume all ECUs have the same counter or can quickly get the correct counter.

The message format for the synchronization limits the size of the counter since we want to transmit the synchronization within one message. For this reason, the natural choice would be a counter of 40 bits. With this counter size and a maximum of 17,543 messages per second [190] that can be sent via the CAN bus, the counter can overflow in about 725.4 days ( $\frac{2^{40}}{17,543 \cdot 60 \cdot 60 \cdot 24}$ ) of continuous communication. An attacker could reduce this by sending even shorter messages by starting a message and sending an error frame immediately. This reduces a message to 15 bits and thus increases the messages per second to 51,459 ( $17,543 \cdot \frac{44}{15}$ ). This can roughly reduce the time till the counter overflows by 3 to only 247.3 days ( $\frac{2^{40}}{51,459 \cdot 60 \cdot 60 \cdot 24}$ ). We consider this value to be unacceptable for the security of our system. For this reason, we transmit a counter value with an extended ID CAN message. This message type introduces 18 more bits to the ID of the message that we utilize to transmit 18 more bits of a counter. A 58-bit counter increases the time till an overflow occurs to 82,884.75 years ( $\frac{2^{58}}{51,459 \cdot 60 \cdot 60 \cdot 24 \cdot 365}$ ). We consider this time span sufficient to protect the counter from an overflow.

---

Due to these decisions, our security protocol is still fully compatible with regular CAN controllers in one network if they only send messages under predefined IDs, which are then not checked by the protocol. Every ECU can read the messages even if it does not speak the BusCount protocol. Furthermore, it is easily adaptable for CAN FD and CAN XL and compatible with the message structure defined by SecOC [8].

### **Practical Challenge**

Because our protocol introduces error frames to invalidate unauthenticated messages, the safety properties of CAN may change. The original mechanism for error frames, where the error counter is incremented by an ECU by 8 if it sends the message or by 1 if it receives the message, is already described in Section 2.1. Important for our change introduced due to the protocol is the question of whether this can result in one or multiple disabled ECUs. From our point of view, only three cases result in this failure. First, an ECU has a technical problem that causes it to send or receive messages incorrectly. Either way, this would result in an error frame since the CRC would be incorrect even without the MAC check. For this reason, it does not affect the safety rating. In the second scenario, an attacker is present that sends unauthenticated messages, causing the error counter to increase, which is similar to an attack against the CRC. Both are not considered in a safety evaluation since these are compliant behaviors. The last scenario concerns the start-up process of a vehicle where different ECUs start communication at different points in time. For this reason, the counter may differ between ECUs. To avoid unnecessary error frames, we suggest that ECUs do not check or process messages until they perform synchronization. This prevents an incorrect counter of ECUs after a wake-up. We conclude that BusCount does not affect the safety of CAN.

### **Performance Evaluation**

It is crucial to ensure that BusCount can be implemented in this setup and that the MAC algorithm is fast enough to check incoming messages within the message transmission. The basic CAN implementation on the ICE40HX8K already requires 4,483 logic cells of the FPGA, leaving only 3,197 remaining logic cells. This limited space does not allow for the classic cipher used for the MAC calculation, such as a CBC-MAC based on AES or an HMAC based on SHA2. Therefore, a lightweight alternative is necessary for our approach. We decided to test Present80 [19] and Prince [21] in CBC-MAC mode, as well as SipHash [4] as an HMAC algorithm. All these MAC algorithms are designed with a

64-bit block size. Since a regular CAN message has at most 29 bits for the ID, 4 bits for the data length, and 64 bits of data, the BusCount approach requires calculating the MAC over 131 bits. These include the 58-bit counter that is not transmitted and subtracts the 24 bits of truncated MAC. Thus, the MAC needs to be calculated over two blocks. Modern alternatives will drastically increase the number of blocks. CAN FD needs 10 blocks for 579 bits of authentic data and CAN XL needs 258 blocks for at most 16,466 bits of data to authenticate. Table 5.2 shows the results of our evaluation of the three ciphers. In the first column, we determine the number of cycles to calculate the MAC for the two blocks. The second shows the total number of logic cells used for the CAN controller.

Table 5.2.: Evaluation of ciphers for secure CAN controller

Algorithm	Cycles	Logic cells (total)
Plain CAN controller	-	4,483
SipHash [4]	13	6,024
Prince [21]	30	5,947
Present80 [19]	68	5,599

The results show each cipher fits on the 7,680 logic cells of the FPGA where Present80 is the smallest and Prince and SipHash have roughly the same size. In terms of speed, the results are the other way around. SipHash clearly outperforms the others with 13 cycles, where Prince needs 30 and Present80 68 cycles for both blocks. Since the calculation of the last block needs to be performed very fast from the moment the last data bit is written to the bus, the calculation needs to be finished before the first bit of the MAC is written. All ciphers were able to do so but if even lower-performance chips were used the slower algorithms could fail. For this reason, we recommend SipHash. The latest crypto analyses [46, 199] also do not show any problems indicating a security risk using SipHash.

## 5.5. Summary on CAN Bus Security

In this chapter, we presented our secure CAN communication protocol, BusCount, to protect an automotive network. In a detailed formal analysis, we compare this solution against the typical characteristics of counter-based approaches currently in use in modern vehicles.

---

We have shown the advantage of our approach in comparison to state-of-the-art protocols. Especially the low-layer message counter, which is decremented with any message start, makes manipulation of the counter impossible for an attacker. The integrated synchronization mechanism allows a secure and fast way to propagate the current counter value for all ECUs in a network with only one CAN message. A further key difference is the fast, low-layer MAC calculation and verification, which enables the verification of the message before it is fully transmitted. This property makes it possible to invalidate a message during the transmission and hinders potential attackers from replaying messages without affecting the system's safety. Thus, the protocol guarantees the freshness of a message since it ensures the authenticity and replay or delay attacks are impossible. In contrast, a generic application-level protocol is vulnerable even if all ECUs have the correct counter, which is not necessarily the case.

Building on the promising results, our future work focuses on implementing BusCount into the firmware of a CAN controller. This step not only enhances the feasibility of our solution but also reduces its costs. Additionally, we are exploring the integration of one or multiple hardware security modules to safeguard the cryptographic keys from potential attackers who might gain access to the controller.

Regarding the attack path feasibility for local attacks causing unintended driving behavior, the presenter technology decreases the probability of bypassing the security mechanisms of the CAN bus to a lower rating. For a replay attack, the time increases through the category  $> 6$  months, resulting in a low attack feasibility.

---

## 6. Automotive Ethernet Security

---

With the widespread adoption of Automotive Ethernet in modern vehicles, the security of in-vehicle communication becomes paramount. Automotive Ethernet is crucial in facilitating inter-domain communication to integrate advanced sensors with ADAS systems and support infotainment systems. As we discussed the importance of securing CAN bus communication in the previous chapter, protecting in-vehicle communication via Automotive Ethernet is equally vital.

In this chapter, we delve into the feasibility of employing Transport Layer Security (TLS) in various communication scenarios, considering different functional and performance requirements. Our analysis builds upon the comprehensive requirements analysis and conceptual framework proposed in [206]. In this paper, I analyzed the requirements and developed the concept of how to introduce TLS into the vehicle network with its different communication protocols. Furthermore, I am responsible for the discussion on the final results of the measurements. This work was the result of the collaboration with two OEMs.

Additionally, Automotive Ethernet enables service-oriented communication through the use of the Scalable service-Oriented MiddlewarE over IP (SOME/IP). In the second part of this chapter, we address the vulnerabilities introduced by this middleware and present our research documented in [203]. In this research, I initially discovered the presented vulnerabilities, implemented corresponding attacks, and performed practical tests against various implementations. Furthermore, I devised two countermeasures and verified the security of the security of SESO-RC. Our work in this area was not only recognized with the Best Paper award of ARES 2021 but also sparked discussions with OEMs and suppliers, leading to the introduction of different security mechanisms to mitigate these vulnerabilities.

We also present the first systematic review of security attacks and countermeasures for Automotive Ethernet in ACM Computing Surveys [189] covering 172 papers, where we collaborated with the National Research Council of Italy. In this work, I coordinated the

---

---

review process of the 172 papers within the Fraunhofer team and reviewed a third of the documents, answering the defined research questions categories of the papers.

To provide context for this chapter, we first give details on the relevant parts of the reference architecture introduced in Section 3.1. These architectural components play a crucial role in the following discussions on Automotive Ethernet security.

## 6.1. Setting of Automotive Ethernet Network

Notably, most connections inside and between domains in classic vehicles have been established using the CAN Bus, which offers a maximum bandwidth of up to 1 Mbit/s. In some cases, other protocols like MOST or FlexRay have been utilized. However, there is a current trend towards faster connections such as CAN FD or Automotive Ethernet. Automotive Ethernet, for instance, can provide speeds of up to 1000 Mbit/s, similar to Fast Ethernet commonly used in LAN networks. With Automotive Ethernet, network topologies shift from a bus system to a star system with switches. According to [69] and [41], Automotive Ethernet is initially employed to interconnect the domain controllers, with the communication gateway acting as the switch connecting all domain controllers. Additionally, certain domains form subnetworks with internal switches that connect each ECU of the domain to the domain controller.

This aligns with our reference architecture introduced in Section 3.1, which serves as the adaptable basis for evaluating the security of Automotive Ethernet communication in modern vehicles. Figure 6.1 illustrates the various uses of Automotive Ethernet in our architecture. It serves as a backend communication system connecting the different domain controllers and facilitates communication within domains, such as the autonomous driving domain or the infotainment domain. Furthermore, most modern vehicles incorporate an Automotive Ethernet channel via the OBD-II connection.

We have previously introduced various communication scenarios in vehicles in Section 3.2. To further analyze IP-based communication, we meticulously consider all these scenarios: internal vehicle communication, including unicast, multicast, inter-domain, and service-oriented communication, as well as diagnostic and external communication.

Our reference architecture utilizes SOME/IP as a service-oriented protocol since it is the AUTOSAR standard. A typical configuration consists of a few switches and a relatively small number of Automotive Ethernet-ECUs, usually in the low two-digit range, with each ECU providing several tens of services, as mentioned in [56].



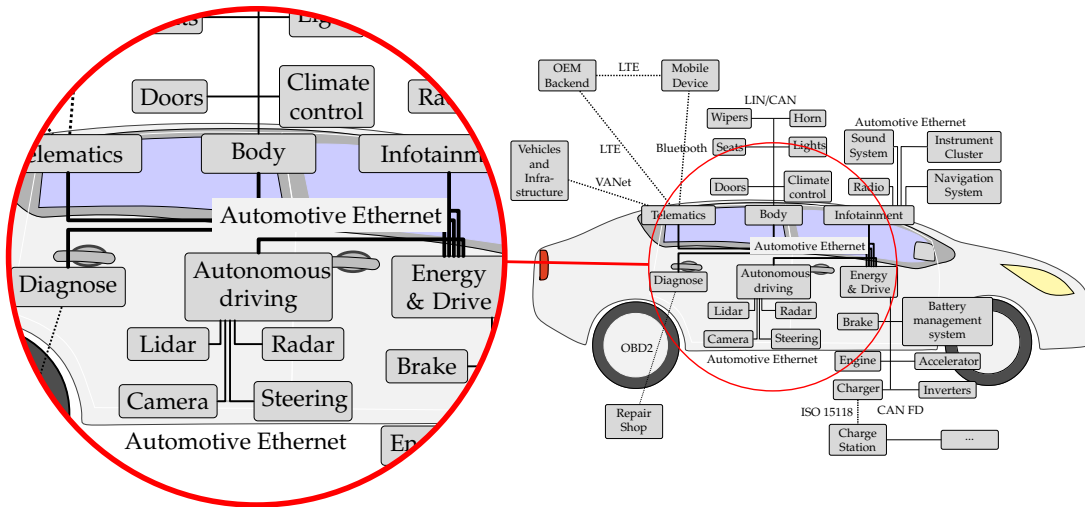


Figure 6.1.: Example of the use of Automotive Ethernet in our Reference Architecture

In our reference network, we assume the presence of around 250 services evenly distributed across 10 ECUs, resulting in an average of 25 services per ECU that implement service-oriented communication. Initially, the ECUs do not possess information regarding the IP addresses or ports of the offered services. Therefore, they need to synchronize during startup or prior to utilizing a specific service.

We anticipate that approximately 20% of the services will be infrequently used and subsequently forgotten after a certain period, typically around 1 minute. Examples of such services include those required for a parking assistance function, where sensor values are intermittently transmitted.

## 6.2. Introducing TLS to In-Vehicle Communication

When considering the potential threats discussed in Section 3.3, securing the communication between ECUs and connected devices becomes crucial to prevent attackers from compromising the vehicle's safety. It is essential to implement mechanisms that ensure data origin authenticity and integrity. Confidentiality may also be required in specific scenarios, such as in product piracy or privacy protection cases.

---

With the increasing adoption of Ethernet and related technologies like IP, UDP, and TCP in modern vehicles, replacing traditional bus technologies like CAN and FlexRay, it is logical to evaluate the suitability of well-established cryptographic protocols like TLS. However, it is important to note that TLS was not originally designed to meet the specific requirements of embedded devices with limited memory and computational power.

This section analyzes whether employing TLS or similar technologies can effectively ensure secure communication between ECUs, external devices, or backend systems when utilizing high-bandwidth technologies like automotive Ethernet. We present three key contributions in this section. First, we conduct a requirement analysis encompassing both real-time and security requirements for communication in modern vehicles. Second, we propose a key management concept and an approach for utilizing TLS to secure in-vehicle communication. Finally, we evaluate the performance of our TLS concept through its implementation on a typical automotive processor.

This section is structured as follows: We begin by analyzing the communication requirements for vehicle networks in Section 6.2.1. In Section 6.2.2, we delve into the application of TLS to secure a vehicle network. Subsequently, we examine whether the performance requirements can be met using TLS on an automotive platform and present the results in Section 6.2.3. In Section 6.2.4, we discuss these results and present some optimization potentials of TLS. Finally, we compare our ideas with related security concepts for vehicle networks in Section 6.2.5.

### **6.2.1. Requirements Analysis**

This section introduces the performance and security requirements that a secure communication protocol for automotive Ethernet must fulfill. First, we describe the performance requirements gathered from multiple sources that are necessary for certain types of communication. These requirements are essential for safety-relevant real-time scenarios that rely on timely information. Second, we introduce security requirements for vehicle communication based on the different communication scenarios of automotive Ethernet introduced in Chapter 2.

#### **Performance Requirements**

Most internal vehicle communication must meet specific performance requirements, which vary depending on the use case. These requirements include service interval, bandwidth,

maximum latency, and maximum packet size. The maximum packet size characterizes the maximum size of a single data package transmitted. The service interval represents the interval between two package transmissions. The bandwidth defines the maximum bits per second transferred via the Ethernet channel. Finally, the maximum latency describes the time between the command to send a package and the reception of this package at the point where the message can be processed. An overview of the performance requirements is shown in Table 6.1, which covers different aspects of vehicle communication, including simple multicast sensor value transfer as well as high-bandwidth video transmission.

The primary source for the requirements of different data types is [117]. This paper introduces several types of communication in a vehicle network, such as control data, driver assistance camera streaming, or video/audio streaming. The communication of sensor values, which was also present in classic CAN or FlexRay communication, is represented by the requirements *Data 1* and *Data 2*. An example of this type of communication could be a multicast transmission of sensor values, such as the vehicle’s current speed or the brake pedal’s position. The difference between *Data 1* and *Data 2* is the stricter latency requirement of 0.1 ms, as suggested in [112]. Furthermore, we added *Data 3*, which includes requirements provided by an OEM. The requirement *Camera* represents an advanced driver assistance system camera, which has an approximate bit rate of 25 Mbit/s for an MPEG2-TS-based video stream. *Audio* and *Video* represent the requirements for audio and video data transmitted for infotainment purposes and, thus, do not have similarly strict maximum latency requirements as the previous data types.

Table 6.1.: Performance Requirements

Data Type	Max. Packet Size [byte]	Service Interval [ms]	Bandwidth [Kbit/s]	Max. Latency [ms]
Data 1 [112]	20	10 - 100	1.6 - 16	0.1
Data 2 [117]	20	10 - 100	1.6 - 16	10
Data 3 <sup>1</sup>	50	1	400	0.5
Camera [117]	786	0.25	25100	45
Audio [117]	1472	8.4	1400	150
Video [117]	1472	1	11800	150

<sup>1</sup>elaborate together with an OEM

---

## Security Requirements

Fulfilling several security requirements is essential to effectively mitigate the risks posed by attackers, as outlined in Section 3.3.

- S<sub>1</sub> Authenticity:* Ensuring communication authenticity is crucial to prevent attackers from injecting malicious messages into the in-vehicle network. For example, verifying the authenticity of a firmware update binary is imperative. Additionally, the identity of communication peers must be verified to prevent unauthorized devices, such as a malicious ECU, from connecting to the electrical system. As mentioned in Chapter 3, appropriate measures should be implemented to prevent attackers from compromising an ECU.
- S<sub>2</sub> Integrity:* Mechanisms for ensuring data integrity are necessary to prevent attackers from tampering with messages undetected. This includes protection against replay attacks, where attackers resend previously transmitted messages. Authenticity and integrity are typically the most critical security requirements that must be upheld in most scenarios.
- S<sub>3</sub> Confidentiality:* Implementing mechanisms to maintain data confidentiality is essential for preventing eavesdropping and unauthorized access by attackers. However, confidentiality may be required only in specific situations, such as protecting intellectual property, personal data, or sensitive information. For instance, firmware updates and advanced diagnostics functionalities may contain proprietary knowledge and should be encrypted. Furthermore, the transfer of personal data needs to be protected from attackers. However, it is crucial to consider the potential downsides of confidentiality measures, such as increased overhead, which can impact real-time communication conditions (cf. Section 6.2.1). Additionally, managing fully encrypted networks may present challenges, requiring data decryption before direct access is possible, even in a garage setting.
- S<sub>4</sub> Availability:* Ensuring continuous and reliable communication is a critical safety requirement. Disruptions in communication can have severe consequences, potentially endangering lives and impacting the functionality of vehicle systems. While this chapter primarily addresses unintentional events rather than direct attacks on availability, it explores the interplay between security measures and the performance requirements discussed in Section 6.2.1. It delves into how security measures can be applied while still meeting the performance criteria.

---

$S_5$  *Freshness*: The security property freshness is important to guarantee that data or messages exchanged between communicating parties are up-to-date and have not been replayed by malicious actors. Achieving freshness is crucial to maintaining the integrity and reliability of communication systems, especially in safety-critical scenarios.

## 6.2.2. Discussion on the Applicability of TLS to Secure In-Vehicle Networks

TLS is the most intuitive solution for securing a vehicle's communication in many scenarios. For example, backend communication can be secured either using HTTPS [173] or a Virtual Private Network (VPN) tunnel [96]. For this reason, it is tempting to apply TLS for further use cases, especially in-vehicle communication.

In the following section, we argue about using TLS for secure in-vehicle communication for the reference architecture described in Section 3.1 and whether it allows us to meet the security requirements specified in Section 6.2.1. For this purpose, we elaborate on how TLS can address the security requirements in every communication scenario. In addition, we elaborate on using a certificate structure inside and outside of a vehicle and describe their validation. The Section 6.2.3 then focuses on our implementation of TLS on a typical embedded controller and the analysis of whether this implementation matches the performance requirements.

### TLS in Different Communication Scenarios

The different communication scenarios have been described in Section 6.1. In this section, we match these scenarios with the possibilities of the TLS protocol. With TLS in place, it ensures the authenticity, integrity, and confidentiality of every message.

**Uni-Cast Communication** Similar to classic direct communication in computer networks, a default TLS connection can also be established between two ECUs. For uni-cast communication in Automotive Ethernet, it is not necessary to change anything in the default TLS protocol. Since TLS is not altered, it fulfills the security properties  $S_1$ ,  $S_2$ ,  $S_3$ , and  $S_5$ , which are discussed in detail in the following security discussion.

---

**Inter-Domain Communication** The communication between domains is pretty similar to the first scenario. The core distinction is that traffic is forwarded through one domain controller into another domain to the ECU. Depending on the setting, the domain controller may need to decrypt the communication and re-encrypt it to forward messages to the next step in the communication chain. However, this poses an additional risk if such a domain controller is compromised because then an attacker has access to all communications that use that domain controller as a relay. Furthermore, additional computational overhead is introduced into the domain controller. Therefore, it is best to establish an end-to-end TLS channel between the communicating ECUs, if possible. The domain controller acts only as a switch and forwards TLS packets between communication partners. In the rare cases that the domain controller itself needs to receive data, this is covered by the first scenario.

**External Communication** Analogous to inter-domain and uni-cast communication, external communication involves a simple communication channel between two entities. However, in this scenario, one communication entity is located outside the vehicle while the other is part of the vehicle itself. As mentioned earlier in this section, TLS has already been proposed to secure external communication between the communication gateway and the external server, such as the OEM backend.

Nevertheless, similar to the inter-domain scenario, if the gateway or domain controller is compromised, it undermines the security provided by the TLS protocol. To address this concern, we propose directly using an additional TLS channel between the ECU and the external entity, ensuring end-to-end security for this critical connection.

Initially, the communication is secured between the gateway and the external entity, preventing unauthorized communication from being forwarded into the vehicle network. Within this TLS channel, an additional TLS channel is established as a second layer of security, directly connecting the external server and the internal ECU. This approach further strengthens the security measures for the communication between the vehicle and the external entity.

**Multicast Communication** Multicast communication is widely used in vehicles to simultaneously transmit messages to multiple ECUs. However, the current TLS standard and typical implementations do not natively support this type of communication. Fortunately, a proposal for multicast TLS based on (D)TLS has been made [98].

In multicast TLS, a group key authenticates and encrypts the multicast messages of every multicast group member. All participants within the group share the same keys, enabling

---

---

them to send and receive messages. This approach presents a trade-off in terms of security. On one hand, it reduces the ability to determine the exact sender of a message. On the other hand, introducing multiple keys (one for each communication pair) would compromise the advantages of multicast messages and result in increased bandwidth consumption.

Using a key for every session would lead to either sending multiple packets with identical content or including the same information multiple times within a single message. In the latter case, each message is encrypted with one key, allowing each receiver to decrypt one of the chunks in the message. While multicast TLS provides a solution for secure multicast communication in vehicles, it is essential to consider the implications and evaluate the specific requirements and trade-offs for each use case before implementation.

## Security Discussion

In addition to addressing the communication scenarios, meeting the necessary security requirements is crucial. TLS offers configurations to provide authenticity ( $S_1$ ), integrity ( $S_2$ ), and confidentiality ( $S_3$ ) [45]. During the initial TLS handshake, the communication partners are authenticated using certificates or pre-shared keys. The record protocol of TLS ensures message authenticity ( $S_1$ ), integrity ( $S_2$ ), and confidentiality ( $S_3$ ) by encrypting messages and appending a Message Authentication Code.

While certificates provide the highest level of security, certain key management issues may require using TLS Pre-Shared Key Cipher suites (TLS-PSK) [181]. TLS-PSK cipher suites come in three types. The first one uses symmetric keys for authentication, the second employs Diffie-Hellman key exchange authenticated with a symmetric pre-shared key, and the third type combines pre-shared key authentication on the client side with public key authentication on the server side.

Regarding using pre-shared keys in the vehicle context, caution is paramount, especially if there is a need for easy ECU exchange without an extensive key-management process. Storing the same key in each ECU type of vehicle model increases vulnerability to attacks, as an attacker who obtains the key from one vehicle can then target all vehicles of the same model. Therefore, the use of TLS-PSK cipher suites should be carefully considered and evaluated.

It is crucial to understand that our focus in this chapter, as mentioned in Section 6.2.1, is not on addressing attacks on availability directly ( $S_4$ ). Instead, we concentrate on examining the interaction between security measures and performance requirements.

---

In scenarios where an attacker gains access to the network, they may attempt to block messages or overload the bus system, resulting in reduced throughput and increased delay. They may also initiate numerous TLS handshakes to overwhelm other ECUs. It is essential to consider additional measures to mitigate these attacks. For instance, in a switched network architecture, an intrusion detection system (as described in [130]) could be deployed to identify attacks and block messages from potentially compromised ECUs or subnetworks.

TLS ensures the correct order of data using sequence numbers for sending and receiving messages. Based on this sequence number, freshness  $S_5$  cannot be guaranteed to be transmitted within a certain period of time, as discussed in the availability section ( $S_4$ ). The freshness  $S_5$  of a new connection is discussed in the following Section, *Certificate Verification*.

### **Proposed Certificate Hierarchy**

For a robust security framework, we advocate the use of certificates over TLS-PSK cipher suites. Each ECU should be equipped with a certificate, serving as either a client or server certificate, to ensure a high level of security.

The proposed certificate should contain the following elements: public key, issuer information, ECU roles and rights, and the signature of a Certification Authority (CA). The corresponding private key should be securely stored only on the ECU itself. The ECU rights information should specify the type of ECU and include a list of rights for accessing other ECUs, data types, or commands. This ensures that ECUs are granted access only to the information necessary for their functionality. Additionally, the receiver can evaluate whether to send data to another ECU and verify if an ECU is allowed to provide certain data or execute specific commands.

We recommend generating public-private key pairs inside each ECU to generate the certificates. During production, the vehicle manufacturer can extract the public key and generate a certificate with all the required fields, which the manufacturer signs. The certificate is then securely written to the ECU. It is crucial to ensure that the private key stays in the ECU to prevent easy compromise during production. To further enhance security, the use of an HSM is proposed for key generation and storage of the private key as a non-exportable key, safeguarding the private key during the ECU's life cycle.

Certificate management necessitates the establishment of a PKI. Figure 6.2 illustrates a possible implementation of a CA hierarchy. The manufacturer operates a long-term root



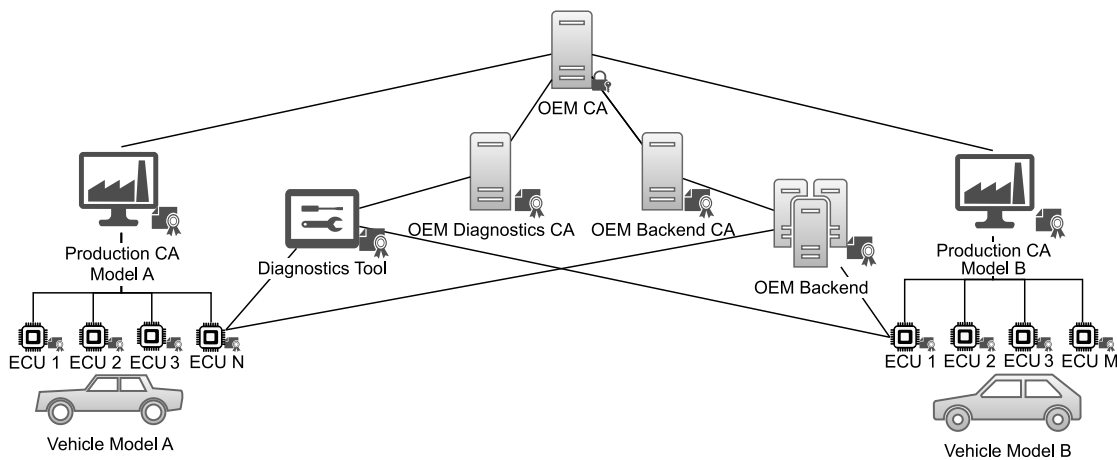


Figure 6.2.: Possible CA Hierarchy

CA named "OEM CA." Below the root CA, several CAs with shorter lifetimes are employed. For each vehicle model, a "Production CA Model X" is used to issue certificates for ECUs specific to that model. If a model is discontinued, the corresponding CA can be disabled. The "OEM Diagnose CA" is responsible for issuing certificates to diagnostic devices that can be used across different model types. An "OEM Backend CA" is utilized to issue certificates for services running in the manufacturer's backend.

### Certificate Verification

Each ECU and diagnosis tool utilizing TLS should be pre-configured with the "OEM CA public key certificate" to enable them to validate certificates. During a TLS handshake, this certificate is used to verify whether the received certificates of the communication partners are indeed issued by the "OEM CA" or a CA lower in the hierarchy. However, two issues regarding certificate verification arise in our scenario.

Firstly, the verification of certificate revocation usually requires an online connection. Typically, a Certificate Revocation List (CRL) hosted on a backend server is checked to verify a certificate. To address this, the manufacturer can operate an Online Certificate Status Protocol (OCSP) Responder, and since modern vehicles are usually online most of the time, CRLs can be downloaded at regular intervals whenever an online connection is available.

---

Secondly, validating the validity period of a certificate relies on the accuracy of the time. In small embedded systems like ECUs, there might not be a precise time source available. Several alternatives can be considered. One solution is to equip each ECU to communicate via TLS with a precise clock and an independent power source to run the clock, although this may incur additional costs. Another option is having a trusted time server within the vehicle that broadcasts the time to the ECUs. This time server can synchronize with an external trusted Network Time Protocol (NTP) server. Alternatively, the validity of a certificate can be tied not to a specific time period but to a certain number of vehicle events. For example, a certificate may only be valid for a certain number of engine starts before requiring renewal. This approach would only require a monotonic counter in an ECU that increments with each engine start and would be compatible with the solution to secure CAN in Chapter 5.

**Certificate Renewal** In the event of a security breach or nearing the end of a certificate's validity period, certificate renewal becomes necessary. Renewal can be performed during maintenance at a trusted garage or repair shop or through the vehicle's online connection.

Renewing certificates from a trusted garage or repair shop during maintenance has the advantage of ensuring the authenticity of the entire installation process. Even if some private CA certificates have been compromised, the vehicle can be updated with new trusted certificates. However, this approach introduces additional overhead during maintenance to update potentially all ECU certificates.

In any case, online certificate renewal may be preferred, especially if no security breach has occurred. It also enables timely renewal of expiring certificates. Valid certificates are used to establish a TLS channel with a backend update server, allowing the installation of new certificates on the ECUs. More detailed concepts for over-the-air updates have been presented, for example, in [146, 148].

### 6.2.3. Performance Analysis

After introducing the security concepts for utilizing TLS in an automotive environment, the next step is to assess the performance of a standard TLS implementation and determine its applicability to a typical automotive platform. To accomplish this, we have established an evaluation environment to measure the performance and subsequently compare our findings with the requirements outlined in Section 6.2.1.

---

---

In the following sections, we present our testbed, which has been specifically designed for this evaluation. We then proceed to showcase the performance measurements, focusing on comparing various cryptographic algorithms. Finally, we discuss the outcomes of our evaluation, explicitly examining the handshake and data transmission aspects of TLS between two ECUs.

## Evaluation Environment

The evaluation environment utilized for our tests comprises two automotive development boards equipped with a TriCore TC297T<sup>2</sup>. This state-of-the-art automotive ECU features three cores, each operating at a clock speed of 300MHz. The platform is equipped with 384KB of EEPROM, 8MB of Flash, 728KB of RAM, and an Ethernet interface capable of supporting speeds up to 100 Mbit/s. The TriCore TC297T is commonly employed in engine control units, electric power steering (EPS) systems, domain controllers, and ADAS. It represents a typical middle-class system from the latest generation and is expected to be utilized in smaller ECUs in the future.

During our evaluations, we exclusively utilized one core of the development board, ensuring that there is ample computational power available to handle the safety-critical functions of the ECU when it is installed in a vehicle.

We established a connection between the two development boards using a 1000 Mbit/s Ethernet switch to eliminate the connection as a potential bottleneck. The switch features a mirroring port, which enables real-time monitoring of the communication between the two evaluation systems without interfering with the communication itself. We connected a control server to this mirroring port to capture and store all messages within the Ethernet network. The server is responsible for initiating and concluding measurements on the development boards.

The software stack employed in the evaluation comprises the real-time operating system ERIKA Enterprise version 2.1.0, integrated with the lwIP TCP/IP stack version 1.4.1. For TLS implementation, we utilized wolfSSL version 3.9.6, a library specifically designed for embedded devices that offers a range of cipher suites. To synchronize the clocks between the boards before each measurement, we employed the PTP. This synchronization ensures

---

<sup>2</sup><https://www.infineon.com/cms/de/product/microcontroller/32-bit-tricore-microcontroller/32-bit-tricore-aurix-tc2xx/aurix-family-tc29xt/sak-tc297t-96f300n-bc/>

---

---

that we can precisely measure the time difference between encryption on one device and decryption on the other.

## Performance Analysis of Cryptographic Algorithms

We conducted the evaluation in two parts: first, analyzing the performance of the cryptographic algorithms themselves, and second, evaluating the performance of TLS. By separating these two aspects, we can exclude the influence of external code, the operating system's scheduling, the delay introduced by the TCP stack, and the communication delay.

In this section, we focus on evaluating asymmetric cryptographic algorithms, which are crucial for an authenticated TLS handshake during the initialization of communication. Subsequently, we analyze the data transfer, which employs symmetric encryption and authentication mechanisms.

**Asymmetric Algorithms** To evaluate the performance of asymmetric algorithms, we selected RSA and the Elliptic Curve Digital Signature Algorithm (ECDSA) (P256), both specified in TLS 1.2 for signing and verification during the TLS handshake. Following the recommendations by [11], we chose a key length of 3072 bits for RSA and 256 bits for ECDSA.

The execution times of the development board for generating a signature and performing verification are displayed in Figure 6.3. RSA can generate signatures within a maximum time of 1.353 seconds in 100 tries, which is relatively slow. However, it exhibits very fast verification, taking only 34 milliseconds in the worst-case measurement. On the other hand, ECDSA performs significantly better in signing data, with an average time of 177 milliseconds in the worst-case scenario. Unfortunately, the verification process takes around 345 milliseconds, which is approximately ten times longer than RSA.

Both signing and verification are necessary to establish a TLS channel between two components. When both operations are required, we recommend using ECDSA over RSA. ECDSA offers advantages such as smaller key sizes and faster execution. However, neither algorithm appears promising for establishing a TLS channel when data needs to be transmitted since the cryptographic functions themselves cannot be executed in under 0.5 seconds. This issue is further discussed in Section 6.2.4. As a result, the performance of symmetric encryption and authentication mechanisms becomes more critical, considering that the TLS handshake can occur in advance.

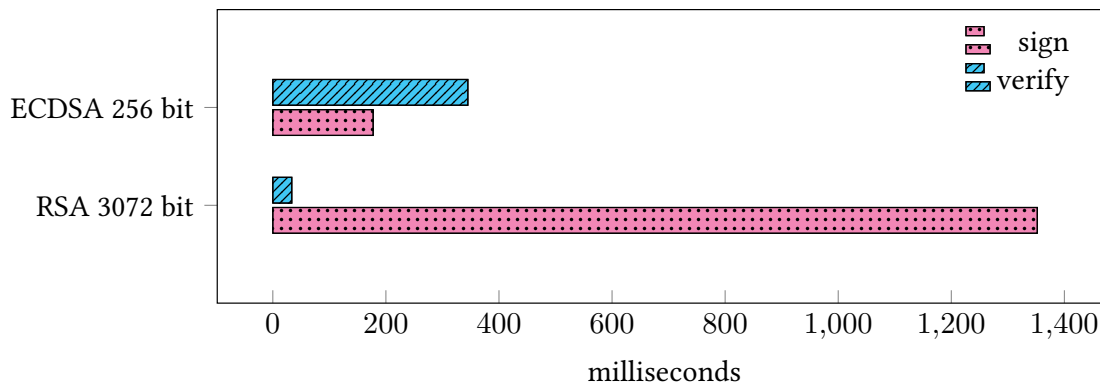


Figure 6.3.: Execution Speed of Asymmetric Algorithms

**Symmetric Algorithms** Unlike asymmetric cryptographic algorithms, symmetric ones perform much faster, making them the go-to for data transportation once a connection is established. Our comprehensive evaluation focuses on assessing the speed of data encryption and authentication on our embedded system. To accomplish an adequate comparison, we test various encryption algorithms combined with authentication procedures and measure the achievable throughput. The results are presented in Figure 6.4.

For our evaluation, we selected commonly used combinations of hash functions, including SHA-1 and SHA-256, and encryption algorithms, namely 3DES, AES in CBC mode, and AES in GCM mode. Our findings indicate that 3DES is the slowest algorithm, and SHA-256 reduces the throughput more significantly than SHA-1. As expected, AES encryption demonstrates faster performance with shorter keys, as 256-bit encryption requires more cycles for encryption compared to AES with a 128-bit key. Additionally, it is interesting to note that GCM mode is slower than CBC, despite CBC mode being combined with a hash-based MAC function.

Furthermore, we explored the throughput achievable with different stream ciphers. The implementations that were provided included ChaCha, Rabbit, and HC-128 combined with Poly1305. These stream ciphers easily outperform the block ciphers quite easily. For instance, ChaCha20 achieves a practical throughput of 20.02 MBit/s, while HC-128 combined with Poly1305 achieves an even higher throughput of 43.75 MBit/s. These results underscore that while encryption does introduce a noticeable bandwidth overhead, it remains viable for real-world scenarios, as outlined in Section 6.2.1, where a maximum throughput of 25.1 MBit/s is required.

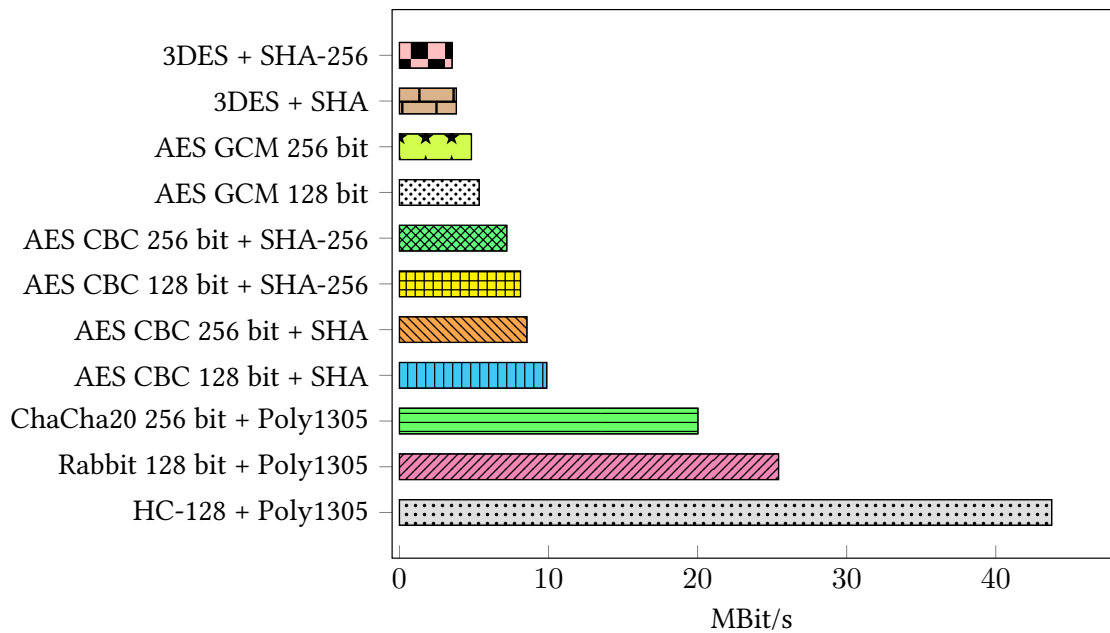


Figure 6.4.: Throughput of Symmetric Ciphers

### TLS Evaluation

The previous measurements provided a general performance assessment of cryptographic operations in our evaluation setup. However, factors like the network behavior and the packet sizes that encryption must handle also influence the performance of a TLS connection.

To evaluate the performance of TLS in a vehicle environment, we measured the communication for different message types and compared the results to the requirements defined in Section 6.2.1. First, we examine the regular handshake process of TLS, followed by an analysis of the symmetric authenticated and encrypted channel.

**Handshake** The TLS handshake initiates every communication channel and is responsible for the key exchange between the communication partners. In this evaluation step, we measured the duration of different parts of the handshake process. As mentioned in paragraph 6.2.3, our previous results indicate that elliptic curve cryptography outperforms

---

---

RSA. Therefore, we measured the performance of Elliptic Curve Diffie-Hellman (ECDHE) combined with ECDSA using a 256-bit key.

For comparison, we also evaluated a handshake with an RSA-signed certificate using a 3072-bit key. Additionally, we focused on handshakes that support perfect forward secrecy, which adds some computational overhead but enhances security in case a symmetric key is compromised. The results of our evaluation are presented in Table 6.2.

Handshake Step	ECDSA 256 bit	RSA 3072 bit
	Time [s]	Time [s]
Client Hello	0.0000	0.0000
Server Hello	0.0004	0.0004
Certificate	0.0758	0.0973
Server Key Exchange	0.3535	1.5635
Server Hello Done	0.5022	1.6470
Client Key Exchange	1.1251	1.9969
Change Cipher Spec / Encrypted Handshake	1.1267	1.9982
Change Cipher Spec / Encrypted Handshake	1.2987	2.1695

Table 6.2.: ECDHE Handshake

Table 6.2 presents the duration of a regular handshake using RSA, which takes considerably longer (2.17 seconds) compared to an ECDSA handshake (1.3 seconds), aligning with our previous measurements. Another noteworthy observation is that ECDSA handshakes show minimal performance variation among the 100 generated handshakes, with the most significant difference being 0.23% from the average result. On the other hand, RSA handshakes exhibit a more extensive performance spread, reaching 2.97%, albeit still relatively low.

It is important to note that the fastest handshake recorded still took more than 1.3 seconds and only involved server authentication. The overall time for a complete handshake, including client certificate verification, exceeds 2 seconds. Clearly, such a lengthy duration is impractical for scenarios where information needs to be transferred within milliseconds. After analyzing the transport channel, we discuss this issue further in ??.

**Transport** Data transmission in the TLS protocol involves symmetric encryption, which is significantly faster than the asymmetric encryption used in the handshake phase. Consequently, it has the potential to meet the requirements outlined in Section 6.2.1. To

evaluate the performance of TLS-encrypted data transmissions, we conducted experiments with different packet sizes, focusing on throughput and delay. The captured encrypted messages were obtained from the mirroring port of the switch, timestamped, and decrypted to extract the packet payload. We calculated the throughput using the payload size and timestamps, comparing the results with the requirements specified in Section 6.2.1.

For our experiments, we examined modern AES cipher suites and the streaming cipher ChaCha20 combined with Poly1305. Additionally, we included two ciphers that solely use HMAC, which is commonly employed in literature to secure CAN bus messages (see Section 6.2.5).

Table 6.3.: Throughput in Mbit/s

	Data 1	Data 2	Data 3	Camera	Audio	Video
Requirements	0.016	0.016	0.4	25.1	1.4	11.8
TCP	0.16	0.16	0.40	33.27	38.72	38.72
NULL + SHA	0.16	0.16	0.40	(11.67)	13.86	13.86
NULL + SHA256	0.16	0.16	0.40	(8.91)	10.26	(10.26)
AES128 + SHA	0.16	0.16	0.40	(5.78)	6.56	(6.56)
AES128 + SHA256	0.16	0.16	0.40	(4.94)	5.59	(5.59)
AES256 + SHA	0.16	0.16	0.40	(5.30)	5.99	(5.99)
AES256 + SHA256	0.16	0.16	0.40	(4.60)	5.19	(5.19)
AES128 GCM	0.16	0.16	0.40	(4.36)	4.58	(4.58)
AES256 GCM	0.16	0.16	0.40	(4.05)	4.28	(4.28)
ChaCha20 + Poly1305	0.16	0.16	0.40	(11.57)	13.09	13.09

The throughput in Mbit/s for each data type with different cipher suites, including a plain TCP connection for reference, is presented in Table 6.3. We observe that for small data packets (Control Data 1, Control Data 2, and Control Data 3) with low throughput requirements, all ciphers can achieve the necessary throughput. However, when larger data packets need to be transmitted, the throughput becomes a challenge. Without encryption, both audio and video data can be transmitted at a rate of 38.72 Mbit/s. With encryption, the throughput is significantly reduced.

The fastest options are the variant with SHA-1-based MAC, reaching 13.86 Mbit/s, and ChaCha20 combined with Poly1305, achieving 13.09 Mbit/s. AES in CBC mode and AES in GCM mode prove to be too slow. Unfortunately, none of the cryptographic algorithms provide sufficient bandwidth to meet the requirement for camera stream data.



To measure the delay, we recorded timestamps before the encryption commenced is executed. After transmitting the message to the second platform, it is decrypted, generating a second timestamp. The difference between both timestamps represents the latency caused by communication via TLS. In contrast to previous measurements, latency is not a concern when transmitting large data packets. Instead, small packets transmitted at high rates may encounter bottlenecks due to encryption. This is evident in the measured results presented in Table 6.4.

Table 6.4.: Latency in ms

	Data 1	Data 2	Data 3	Camera	Audio	Video
Requirements	0.1	10	0.5	45	150	150
TCP	(0.192)	0.192	0.213	0.277	0.310	0.310
NULL + SHA	(0.319)	0.319	0.378	1.128	1.791	1.791
NULL + SHA256	(0.419)	0.419	0.484	1.473	2.353	2.353
AES128 + SHA	(0.517)	0.517	(0.602)	2.285	3.683	3.683
AES128 + SHA256	(0.635)	0.635	(0.742)	2.453	3.982	3.982
AES256 + SHA	(0.530)	0.530	(0.625)	2.453	3.982	3.982
AES256 + SHA256	(0.653)	0.653	(0.785)	2.850	4.467	4.467
AES128 GCM	(0.332)	0.332	0.441	2.952	4.134	4.134
AES256 GCM	(0.347)	0.347	0.456	3.147	5.428	5.428
ChaCha20 + Poly1305	(0.251)	0.251	0.278	1.153	1.889	1.889

Table 6.4 displays the latency in milliseconds for each data type. The communication scenarios media data, including audio and video, demonstrate negligible latency across all algorithms. However, Control Data 1 and Control Data 3, which involve significantly smaller packet sizes and require lower latencies, present challenges. Even plain communication fails to meet the requirement of 0.1 ms latency for Control Data 1. Except for AES in CBC mode, all ciphers achieve a latency below 0.5 ms for a packet size of 50 bytes, as required for Control Data 3. Additionally, it is noteworthy that GCM mode encryption outperforms encryption with CBC mode combined with a hash algorithm for small data packets, as the hash algorithm substantially increases the packet size. Conversely, CBC mode encryption is faster than GCM when dealing with larger messages, as the overhead added by the hash algorithms becomes less significant. Overall, cipher suites with ChaCha20 perform best in every scenario.

---

## 6.2.4. Discussion of Measured Results

The measurements conducted in this study reveal that TLS can effectively meet major requirements even within vehicular environments. The previous section demonstrated that, in terms of performance, the ChaCha20 + Poly1305 cipher suite exhibits significant potential and is currently included in the TLS standard. Unlike AES in GCM or CBC mode, which is sensitive to packet size, the performance of ChaCha20 + Poly1305 remains consistent. Furthermore, this cipher suite ensures confidentiality, in contrast to NULL+SHA. In this section, we address the remaining issues identified during our evaluation. Firstly, we explore various approaches for performing handshakes to establish TLS connections. Subsequently, we examine the performance limitations of encrypted channels in two specific use cases.

### Optimizing the TLS Handshake

The measurements in Section 6.2.3 revealed that the standard handshake procedure between two embedded devices, including client and server certificate exchange, takes more than two seconds to complete. In the context of vehicle networks with 50 or more ECUs, where hundreds of connections need to be established, the initialization time for each connection becomes a significant concern. Even with the possibility of parallelizing key exchanges to some extent, the time required to initialize connections for a single ECU can easily be half a minute or more if the most connected ECU requires 20 connections. Clearly, it is impractical to have such lengthy handshake procedures during the start of communication or engine start. Drivers would not be willing to wait a minute or longer before the engine can start. Therefore, reducing the time required to perform a handshake is crucial.

One potential solution is using pre-shared keys (PSK), significantly reducing the handshake time to less than 100 ms. However, it is important to note that key establishment using pre-shared keys is less secure, as discussed in Section 6.2.2. Another interesting approach is to combine asymmetric ciphers with pre-shared keys, as proposed in the 0-RTT (Zero Round Trip) TLS protocol [179]. However, the 0-RTT construct has potential security flaws, as highlighted in [57]. To address these issues, TLS 1.3 [152] introduces a set of recommendations to mitigate replay attacks in 0-RTT, such as "Single-Use Tickets" and "Client Hello Recording", which need to be present to ensure replay protection.

Another avenue to explore is improving the performance of Elliptic Curve Cryptography (ECC). Existing literature offers several examples of performance enhancements for ECC,

---

such as the work presented in [49], which demonstrates the possibility of achieving a ten-fold performance increase for ECC in embedded systems compared to other implementations. By leveraging such improvements, an ECC handshake can become comparable in speed to the tested pre-shared key implementation. However, it is essential to consider that the simultaneous establishment of multiple connections consumes bandwidth and may increase the latency of handshakes.

To address this challenge, we propose a solution inspired by [135], which involves separating the authentication process from the actual communication. We suggest that handshakes can be performed during an ECU's idle time, such as when a vehicle is stopped or when the vehicle is in the process of pre-heating the interior. During these idle periods, handshakes can be executed gradually, starting with older session keys. In the event that the vehicle starts before every connection establishes new session keys, older session tickets can be stored on the client and server sides. These stored session tickets can be utilized to resume connections without the need for a handshake. This approach allows for the renewal of session keys within a reasonable time frame while ensuring that existing connections can be quickly reestablished for subsequent drives.

By implementing these optimizations and separating the authentication process from the communication phase, we can significantly reduce the time required for TLS handshakes, making them more practical and efficient for use within vehicles.

### **Optimizing Performance of the Encryption Channel**

Section 6.2.3 highlights that the use cases "Control Data 1" and "Camera" fail to meet the required performance criteria.

For the "Control Data 1" use case, a latency of 0.1 ms was specified, whereas the fastest encrypted channel (ChaCha20) requires 0.251 ms. A comparison with plain TCP connections revealed that even TCP falls short of the required speed (0.192 ms). Interestingly, when subtracting the TCP overhead from the encrypted connection, it becomes evident that the encryption and decryption process only takes 0.059 ms. Thus, the majority of time is consumed by TCP communication rather than the encryption itself. By switching the communication protocol to UDP, for instance, achieving the desired performance level and meeting the specified requirements is possible.

The second challenge pertains to the throughput of the camera use case. The desired throughput is 25.1 Mbit/s, which can be achieved with TCP (up to 33.27 Mbit/s on our development platform). However, ChaCha20 falls short, reaching only 11.57 Mbit/s. Even

---

without considering the limitations of the transport channel, ChaCha20 can only reach a maximum throughput of 20.02 Mbit/s. To meet the target throughput, it is necessary to employ faster cipher streams like HC-128 or Rabbit, which are not currently part of TLS. With HC-128, throughputs of 43.75 Mbit/s (without the TCP communication) can be achieved. Additionally, enhancing processor performance by utilizing multiple CPU cores can lead to faster connections, as our measurements were conducted on a single core.

By considering alternative protocols and exploring faster cipher streams, along with optimizing CPU utilization, the performance issues encountered in the "Control Data 1" and "Camera" use cases can be addressed.

### **6.2.5. Related Work on Securing Automotive Ethernet**

Security in in-vehicle networks has been extensively studied in both literature and standardization efforts. In this section, we provide an overview of previous work on vehicle network security, which has often focused on addressing the limitations of CAN bus systems.

Many studies on secure communication in vehicle networks emphasize the importance of authenticated encryption, achieved by sending a MAC with or after the messages. These have been extensively discussed in Section 5.1.2 and are not iterated here again. Most trade security for the ability to fit into small CAN packages and focus solemnly on authenticity. In contrast, TLS in Automotive Ethernet offers a state-of-the-art security level.

Key distribution has also been a topic of discussion in in-vehicle network security. Some proposals introduced additional devices dedicated to key distribution or firmware integrity monitoring tasks. While TLS introduces overhead in ECU communication and consumes more computational power, it provides confidentiality through encryption algorithms and authenticity through a public key infrastructure.

Schulze et al. [163] suggested a Data Management System (DMS), a central data storage system that receives data from various ECUs instead of directly communicating with ECUs. The DMS enforces access control for reading or updating data on this central device, along with ensuring data integrity. It also allows for protected storage in the event of an accident.

---

Groll and Ruland [64] proposed the concept of grouping ECUs into trusted groups, where ECUs within the same group share a symmetric key. Key management is performed by a Key Distribution Centre (KDC) within the vehicle network. The KDC distributes symmetric keys to each ECU, and with signed Access Control Lists (ACLs), each ECU can prove its membership in trusted groups to the KDC. Asymmetric encryption between the ECU and KDC protects key distribution for trusted group keys.

Oguma et al. [139] suggested adding integrity validation of ECU software through hash functions. Symmetric key exchange for ECU communication encryption is only performed if the validation of hash values is successful. Their architecture involves a central ECU that attests to the correct software state of every ECU, maintaining a list of all hash values. Validated ECUs receive symmetric keys via a Key Predistribution System (KPS) and can encrypt and sign messages using these keys. Each message includes replay protection and proof of ECU integrity.

Wolf et al. [197] presented a similar approach to enhance communication security in a vehicle network. In their architecture, each ECU is required to authenticate itself to a gateway using a certificate. The gateway distributes symmetric encryption keys to authenticated ECUs, and this key is shared within the local network. The key is used to encrypt and authenticate each message in the local domain.

In the Internet of Things (IoT) context, Raza et al. [151] suggested using Internet Protocol Security (IPsec) to provide authentication, encryption, and integrity checks. Several studies have also proposed secure communication solutions for IoT. Caposelle et al. [30] presented a framework that optimized DTLS for low-end devices, considering constraints such as small code size and minimal power consumption.

### **6.3. Secure Service Oriented Automotive Communication**

In the previous section, we focused on the applicability of TLS in static communication scenarios typical for classic vehicles. However, introducing Automotive Ethernet and IP in vehicles enables dynamic and service-oriented communication with efficient bandwidth utilization.

One prominent middleware solution for service-oriented communication in automotive systems is the Scalable service-Oriented MiddlewarE over IP (SOME/IP), which supports remote procedure calls, event notifications, and serialization/wire format (as introduced in Section 2.2.1). Developed specifically to meet automotive requirements, such as

---

integration in AUTOSAR and fast response times, SOME/IP can be utilized by ECUs of different sizes and operating systems. ECUs can announce offered services as servers or make use of services as clients through SOME/IP.

Unfortunately, the SOME/IP specification does not incorporate any security mechanisms. As a result, it is often combined with link-layer security protocols like Media Access Control Security (MACsec) or AUTOSAR SecOC. In this section, we conduct a formal security analysis of SOME/IP using the Tamarin prover<sup>3</sup> to investigate its security implications, particularly focusing on MitM attacks. We discover these attacks are feasible even when link layer security protocols are in use if an attacker compromises a single ECU in the vehicle (as described in our attacker model in Section 3.3). The MitM attack allows the intruder to impersonate a SOME/IP server and a client, redirecting the communication through the attacker. To validate our findings, we implement these attacks and successfully target the open-source reference implementation `vsomeip` [18] and the automotive development and testing tool `CANoe` [63].

To address the security vulnerabilities introduced by the service discovery of SOME/IP, we propose two protocol extensions that ensure a protected service discovery process and secure subsequent data transmissions. Our solutions enable authorization and authentication for SOME/IP services. The first approach utilizes classic asymmetric cryptography, employing certificates, digital signatures, and Diffie-Hellman key exchange to establish symmetric keys that protect the subsequent SOME/IP communication. The second protocol extension is specifically designed for resource-constrained embedded devices and leverages efficient symmetric cryptography. However, an additional authorization server (AS) is required within the E/E architecture. Both approaches mitigate the impact of an attack resulting from a compromised ECU by securing service provisioning and usage. We use the Tamarin prover to formally verify the security properties of both proposed protocol extensions and evaluate their performance and bandwidth overhead.

This section is structured as follows: In Section 6.3.1, we discuss the possible security mechanisms that can be implemented in a vehicle network. In Section 6.3.2, we review related work on securing SOME/IP and other service-oriented communication protocols. An overview of SOME/IP is provided in Section 2.2.1. We introduce our formal security analysis approach in Section 6.3.3, which we utilize to identify the MitM attacks described in Section 6.3.4. The implementation and evaluation of these attacks are described in Section 6.3.5. In Section 6.3.6, we present our two security extensions for SOME/IP, detailing their formal analysis and practical evaluation.

---

<sup>3</sup><https://tamarin-prover.github.io/>

---

### 6.3.1. Network Security Assumptions

Since SOME/IP lacks inherent security mechanisms, it is vulnerable to attacks. However, to analyze SOME/IP under realistic conditions, we assume the presence of network security protocols such as MACsec [79, 80], the previously discussed TLS, or AUTOSAR's SecOC [8]. Consequently, ECUs only accept messages from other ECUs that can be authenticated.

MACsec, standardized as IEEE 802.1AE, is utilized to secure layer 2 Ethernet networks. It provides data integrity, data origin authentication, and optional data confidentiality. The default cipher suite is GCM-AES-128, which employs AES in Galois/Counter Mode with a 128-bit key. Additionally, GCM-AES-256, utilizing a 256-bit key, can be selected. MACsec Key Agreement (MKA), defined in IEEE 802.1X-2010, facilitates key establishment between nodes through a key server. MACsec has been proposed as a security measure for Automotive Ethernet networks, as discussed in [33] and [165].

SecOC, standardized in AUTOSAR, ensures the security of messages transmitted over an automotive communication bus. SecOC guarantees data integrity and data origin authentication by appending a truncated MAC and a freshness value to each message.

### 6.3.2. Related Work on Securing Service-oriented Communication

In the context of in-vehicle networks utilizing classical automotive bus systems like CAN, complex MitM attacks were not considered relevant due to the necessity of routing in a bus system. Previous research on in-vehicle attacks primarily focused on replay or injection attacks on CAN and other bus systems. These attacks have been introduced in Section 3.3.

Several approaches for secure in-vehicle communication have been given in the literature. AUTOSAR standardizes SecOC [8] for securing CAN communication using symmetric cryptography. The security of SecOC has been formally analyzed by us in [110] using the Tamarin prover. With the increasing computing power and higher bandwidth available, asymmetric cryptography can also be employed. TLS, which is part of both AUTOSAR's classic and adaptive platforms, is one example of this. The protocol has been suggested first by us in [206] as discussed in Section 6.2 for vehicle networks. IPsec is supported by the adaptive platform as well. Formal analyses using tools like Tamarin [38] and ProVerif [15] have been conducted on TLS 1.3. Tamarin has also been utilized to analyze V2X revocation protocols [194] and an electric vehicle charging protocol [115].

In classical Ethernet networks, MitM attacks are well-discussed. Protocols such as Address Resolution Protocol (ARP), Spanning Tree Protocol (STP), Dynamic Host Configuration

---

Protocol (DHCP), and Domain Name System (DNS) are vulnerable to MitM attacks, allowing attackers to gain a MitM position and perform further malicious actions such as message redirection, modification, or eavesdropping. Countermeasures like MACsec, IPsec, and TLS have been extensively studied in the literature, as shown in [103].

Securing SOME/IP is discussed in [108], proposing a central entity for key material distribution using (D)TLS during the event group subscription. However, the `service offer` and `find offer` messages are not protected. SOME/IP itself was extended with a TESLA protocol to allow broadcast communication. Using TESLA with delayed authentication for broadcast messages, as described in [144], requires modifications to the SOME/IP process flow and introduces latency that may not be suitable for time-critical data.

To detect attacks on SOME/IP, Herold et al. introduced an Intrusion Detection System (IDS) based on Complex Event Processing (CEP) in [76]. This IDS can identify attacks in the form of malformed packages, protocol violations, and timing issues. However, like other IDS approaches, it suffers from false positives and false negatives.

Iorio et al. propose a security framework for protecting SOME/IP communications in [84]. They suggest performing individual handshakes between each client and the service-offering server using certificates and asymmetric cryptography to establish a shared symmetric key for securing subsequent messages. However, similar to TLS, this approach introduces overhead due to asymmetric cryptography. Moreover, it is not designed to broadcast service offer messages. Instead, `FindService` messages are sent to perform individual key exchanges.

The discussion on securing service-oriented communication extends to other areas as well. For instance, in order to secure the Robot Operating System (ROS), the Secure ROS fork [175] enables secure communication among ROS nodes by establishing an IPsec connection and restricting service providers and requesters to predefined allowed IP addresses. By coupling IP addresses and permissions, the idea of a dynamic network with distributed services over different entities is no longer possible.

### 6.3.3. Formal Security Analysis Approach

In this section, we present our approach to formally analyze the security of SOME/IP in combination with a secure channel and our two proposed security extensions for SOME/IP. We utilize the symbolic model, also known as the Dolev-Yao model, for performing the security analysis. This model assumes cryptographic primitives to be perfectly secure,



---

making it suitable for assessing the composition of these primitives and identifying practical attacks on protocols (cf. Section 6.3.2).

As SOME/IP lacks inherent security mechanisms, we narrow the capabilities of the attacker by assuming the use of an underlying authentication protocol like SecOC, TLS, or MACsec. However, we still consider the possibility of the attacker compromising ECUs as per the *Advanced Remote Attacker* proposed in our attacker model (cf. Section 3.3).

To analyze our proposed security extensions, we employ the same powerful Dolev-Yao attacker model, which assumes complete control over the network, including the ability to compromise ECUs. This allows us to thoroughly evaluate the effectiveness of our extensions under worst-case scenarios.

### Security Properties in the Symbolic Model

In our analysis, we focus on authentication to prevent unauthorized manipulation of SOME/IP communication, particularly in the context of MitM attacks. To assess authentication, we employ the authentication properties defined by Lowe [119], which form a hierarchy ranging from the relatively weak *aliveness* property to the more robust *injective agreement* property.

The *aliveness* property, applied to a message authentication protocol, ensures that when a message is received and accepted by a protocol entity, it can be inferred that the entity presumed to be the sender had previously sent a message. However, the received message does not have to be identical to the original message. This property verifies the "aliveness" of the sender, confirming its previous engagement in the same protocol with the corresponding role.

We utilize the *injective agreement* property to verify our proposed security extensions (cf. Section 6.3.6). This property mandates that when a message is received and accepted, an identical message has been previously sent to the recipient by the claimed sender. Additionally, the recipient can only accept the message once, making the message replay by the attacker infeasible. Thus, the *injective agreement* serves as an effective countermeasure against MitM attacks.

While *injective agreement* is a strong authentication property, it does not directly account for broadcast communication patterns. For protocols that involve broadcast communication in a shared-key setting, such as SecOC (analyzed by Lauser et al. [110]), an adaptation called a *One-time group agreement* is proposed. If the property is fulfilled, a message

---

that is received and accepted was sent by a legitimate sender before. Furthermore, the message is accepted only once by each recipient. Although slightly weaker than *injective agreement*, as it does not verify recipient intent or individual senders, it is suitable for analyzing broadcast communication scenarios and provides reasonable guarantees even in the presence of compromised protocol entities.

In addition, we enforce *syntactic secrecy* for the secrets used within our proposed extensions. This property ensures no valid traces in the model where the attacker learns the secret. Furthermore, *forward secrecy* extends the previous requirement. It ensures that even if the long-term credentials of one or more involved entities are compromised after the protocol runs, the session secrets remain unknown to the attacker.

We utilize the well-established Tamarin prover in the symbolic model to facilitate our formal analysis in Section 5.2.1. This powerful tool automatically proves security properties based on a model of the protocol for an unbounded number of entities or generates a counterexample.

#### 6.3.4. SOME/IP Analysis and Attacks

We assume some secure communication channel is in place to protect SOME/IP. To analyze the security of SOME/IP we use the Tamarin prover. The complete formal model is given in Appendix A.2.1.

We focus especially on the service discovery messages, which are assumed to be secured via group authenticated channels with replay and integrity protection. In the absence of compromised legitimate ECUs, we can establish strong authentication properties, including *One-time group agreement*. Under this assumption, the attacker is unable to inject valid `OfferService` messages.

However, message authentication is no longer guaranteed when considering a stronger attacker capable of compromising an ECU, even with the relatively weak *aliveness* property, which is implied by all other authentication properties we consider (cf. Section 6.3.3). We define the *aliveness* property with respect to `OfferService` as follows:

**Definition 1** (Aliveness w.r.t. `OfferService`). *For an honest server  $S$  and an honest client  $C$ , we require that whenever  $C$  accepts an `OfferService` message  $m$  as coming from  $S$ ,  $S$  has previously been active.*

In Tamarin's syntax, the lemma representing this property is as follows:

---

```

lemma AlivenessOfferService :
  " All server client nonce #i .
  E_C_ReceiveOfferService (client , server , nonce) @i
  ==> (Ex #j . E_S_Created (server) @j)
  | (Ex id #r . E_Corrupted (id) @r
  & Honest (id) @i) "

```

Here, `E_C_ReceiveOfferService` represents the event of the client receiving an `OfferService` message, supposedly from the server. `E_S_Created` denotes the event of the server being created by a rule, indicating its previous activity. Other lemmas use the `nonce` parameter to differentiate between individual `OfferService` messages. The last two lines ensure that neither the server nor the client has been compromised.

The *aliveness* property fails to hold because the attacker has the ability to forge messages from any Service ID. In SOME/IP, there is no mechanism that associates the Service ID within an `OfferService` message with its legitimate sender. Similarly, the Client ID used in service requests is not bound to a specific client. Therefore, when sending a request from a compromised ECU, the attacker can use any Client ID of their choice.

By exploiting these two vulnerabilities, the attacker can execute a MitM attack with minimal prior knowledge of the system. In the following section, we provide a description of these practical MitM attacks that we derived from our formal analysis.

### Copycat Attack on the Service Offer

The general idea of the copycat attack is that the attacker *A* offers a service as soon as a regular server *S* broadcasts its service offer message with the endpoint option of the attacker *A*.

We explain this attack using an example scenario (illustrated in Figure 6.5). Initially, the legitimate server *S* offers a service by sending `OfferService(0x1234, 0x5678)` with Service ID `0x1234`, Instance ID `0x5678`, and endpoint option with its IP address and port `Endpoint(10.0.0.2:30509)`. As soon as attacker *A* receives this message, she sends an identical service offer but with her own endpoint `Endpoint(10.0.0.4:30510)`. Typically, a client *C* would receive both service offers, as the attacker cannot influence lower-layer communication. The client *C* may choose the attacker *A* as the service provider for sending requests. Consequently, *A* can forward these requests to the original server *S* and relay the responses from *S* back to *C*. In this MitM position, *A* can carry out additional attacks such as message manipulation or selective message dropping.

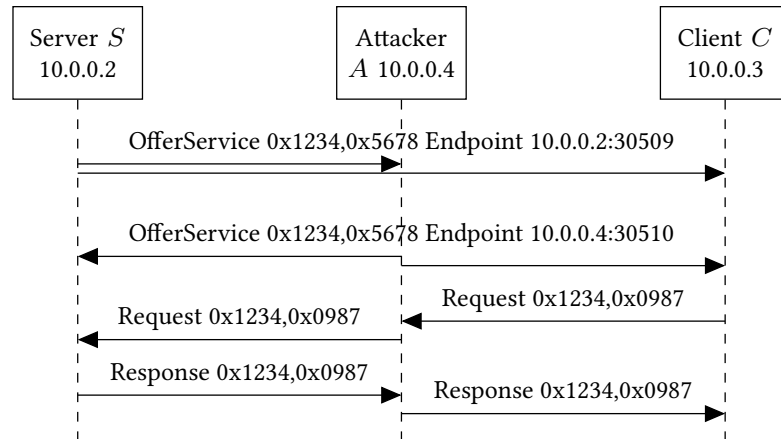


Figure 6.5.: Copycat Attack on the Service Offer

The presence of this attack in our Tamarin model can be demonstrated by a simple verification. For this verification, we try to prove that whenever a client receives a response with generated random data, unknown to the attacker before the send operation, from a server, then the response is sent by the server to the correct client endpoint. Tamarin generates an attack trace that corresponds to the described attack for this verification attempt.

In practice, the copycat attack has several limitations from the attacker’s perspective. Firstly, a client may already be connected to the legitimate server and not switch to another server. Secondly, even if the client is open to changing service providers, there is no guarantee that the client will select the attacker as the service provider. This attack has a fundamental problem depending on the implementation because the client will always receive the service offer from the original server first and the attacker’s service offer second. Furthermore, this approach conflicts with the SOME/IP specification, which requires unique SOME/IP service instances. Therefore, the attack could potentially be detected by an Intrusion Detection System (IDS) could potentially detect the attack. However, as our implementation and evaluation have demonstrated (cf. Section 6.3.5), regular servers simply ignore such messages.

As an alternative, the attacker could try an alternate Instance ID for the same Service ID. If the client accepts service instances other than the original one, it may choose the attacker as the service provider. However, there is still no guarantee that the client will select the attacker as the service instance. Additionally, the attacker needs knowledge of

---

the valid Instance IDs accepted by the client, assuming that the client does not accept all Instance IDs.

In cases where load balancing options are used (cf. Section 2.2.1), the attacker may send a service offer with the highest priority and weight option. If no other service instance with the highest priority and weight option is available, the client may choose the attacker. Of course, multiple honest entities may have the highest priority and weight, leaving the attacker at least with a chance of being picked.

### **De-association Attack on the Service Offer**

To address the limitations of the copycat attack, we have enhanced the attack by deceiving clients  $C$  into believing that the legitimate server  $S$  no longer offers the requested service. The attacker  $A$  accomplishes this by sending an additional unicast message to  $C$  to de-associate  $C$  from  $S$ . This message contains a `StopOffer` with the endpoint option of  $S$ .

An exemplary attack is illustrated in Figure 6.6. The communication begins with  $S$  sending an `OfferService(0x1234, 0x5678)` with `Endpoint(10.0.0.2:30509)`. In response,  $A$  sends a unicast `StopOffer(0x1234, 0x5678)` message to  $C$ , containing `Endpoint(10.0.0.2:30509)`. This leads  $C$  to believe that  $S$  no longer offers the service. Similar to the copycat attack,  $A$  offers the same service herself in order to establish the MitM position.

Tamarin can generate this attack by further restricting the lemma mentioned in the previous section to cases where the client already knew the server beforehand.

The attacker sends the unicast `StopOffer` messages to the clients. If server  $S$  were to receive such a message as well, it would assume that another server has taken over and stopped providing the service, potentially for load-balancing reasons. This would prevent  $A$  from gaining a MitM position. The `OfferService` message from the attacker can be sent as a broadcast since  $S$  simply ignores this message.

The de-association attack requires attacker  $A$  to know which clients are interested in the service in order to send the unicast `StopOffer` messages to them. Identifying these clients poses a challenge for  $A$  since  $A$  can only eavesdrop on broadcast messages. One option for her is to listen to broadcast messages to identify clients and their IP addresses, in particular, the `FindService` message of `SOME/IP` reveals this information. The other

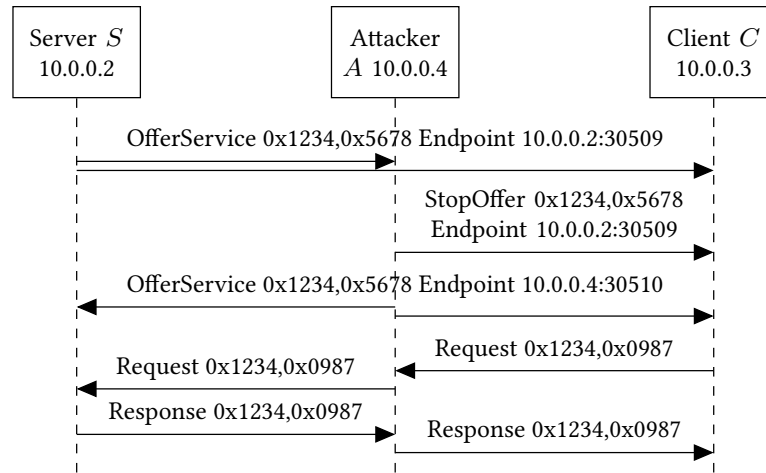


Figure 6.6.: De-association Attack on the Service Offer

option for *A* is to send the `StopOffer` message to the entire IP range of the network (excluding the regular server) to cover all potential clients.

### Attack on Publish/Subscribe

Building on the request/response attack discovered with Tamarin, we have developed an attack on the publish/subscribe communication pattern of SOME/IP, as illustrated in Figure 6.7.

The attack scenario unfolds as follows: The regular server *S* sends an initial service offer message `OfferService(0x1234, 0x5678)` with `Endpoint(10.0.0.2:30509)`. After this message, both the client *C* and the attacker *A* subscribe to the service by sending `SubscribeEventgroup(0x1234, 0x5678)` messages with their respective endpoint options (`Endpoint(10.0.0.3:3333)` for *C* and `Endpoint(10.0.0.4:4444)` for *A*). The server *S* acknowledges these subscriptions by sending the acknowledgment message `SubscribeEventgroupACK(0x1234, 0x5678)`. Furthermore, *A* quickly executes one of the attacks on the service offer, misleading *C* into believing that the attacker now offers the desired service. This results in *C* sending a subscribe event group message `SubscribeEventgroup(0x1234, 0x5678)` with `Endpoint(10.0.0.4:3334)` to subscribe to *A*. *A* sends a `StopSubscribeEventgroup(0x1234, 0x5678)` message

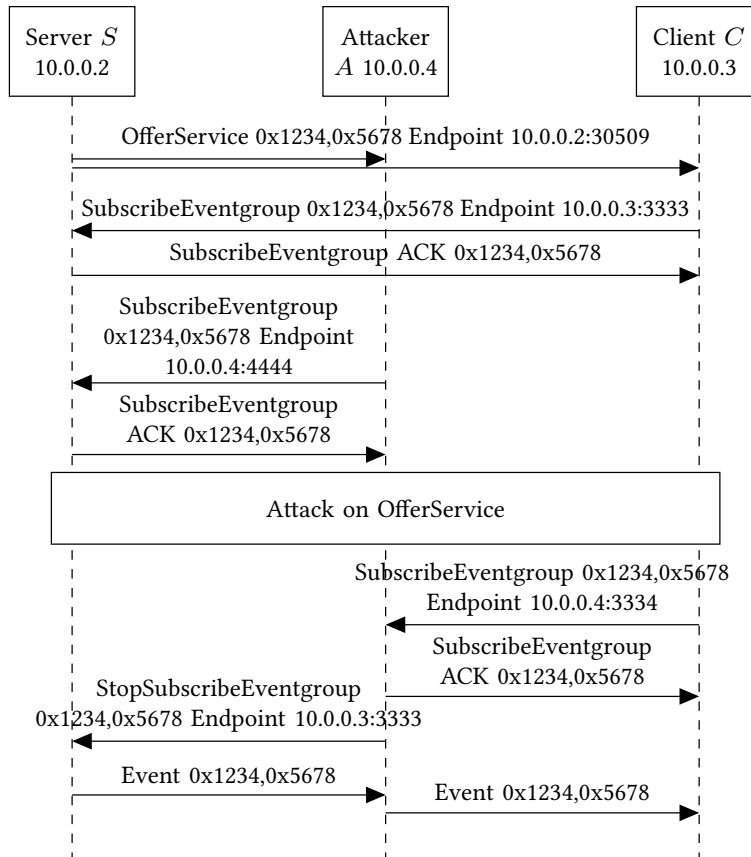


Figure 6.7.: Attack on Publish/Subscribe

to  $S$ , using  $C$ 's endpoint option ( $\text{Endpoint}(10.0.0.3:3333)$ ), effectively unsubscribing  $C$  and preventing  $S$  from sending event messages to  $C$ .  $A$  then sends an acknowledgment  $\text{SubscribeEventgroupACK}(0x1234, 0x5678)$  to  $C$ .  $C$  is now in a MitM position. When  $S$  sends event messages  $\text{Event}(0x1234, 0x5678)$  for its service, these messages are intercepted by attacker  $A$  instead of reaching  $C$ .  $A$  can then manipulate these messages before forwarding them to  $C$  or choose to drop them entirely. This attack compromises the integrity and confidentiality of the event messages, as they can be altered or withheld by attacker  $A$ .

---

### 6.3.5. Attack Evaluation

In this section, we provide a brief overview of our attack implementation, as well as the evaluation process against the libraries `vsomeip` and `CANoe`.

#### Attack Setup

For our first evaluation, we utilized three components: a client, a server offering a service, and the MitM attacker. Client and server are ARM1176JZF-S boards running Linux Debian (Kernel 4.19) and `vsomeip` 3.1.16.1. These components were connected via a switch and communicated using UDP (or TCP). We also evaluated TCP, which yielded equivalent results. We leveraged the examples provided with `vsomeip` as test cases for request/response and publish/subscribe communication. The server offered a service for approximately 10 seconds, regularly sending `OfferService` messages (at most every 2.5 seconds). After 10 seconds, the server sent a `StopOffer` message and waited around 10 seconds before repeating the process. The client sent data requests approximately every second while the service was available. It is important to note that slight variations in message transmission may occur based on system and network load. The publish/subscribe example followed a similar pattern, with the client sending a subscribe message as soon as the service became available. The server then sends events to the subscribed client every second. The attacker component was implemented using `Scapy` [16], which already supports `SOME/IP`.

In our second setup, we employed a VN5610 Automotive Ethernet interface connected to a PC running `CANoe` 9.0.137 [63] with the `SOME/IP` library. The `SOME/IP` client and server were implemented on the PC, while the attacker was implemented in a Linux virtual machine on another machine. The attacker's PC and VN5610 were connected via a switch. The `SOME/IP` communication was implemented analogously to the `vsomeip` implementation. For this experiment, we utilized the same attack implementation as described in the previous case.

#### Evaluation of Service Offer Attacks

Both attacks on the service offer procedure, copycat (cf. Section 6.3.4) and de-association attack (cf. Section 6.3.4), aim to route messages through the attacker. To evaluate the



effectiveness of these attacks, we conducted experiments using both SOME/IP implementations (vsomeip and CANoe). We measured the number of response messages that were sent directly from the server to the client and the number of messages redirected to the attacker.

The results of our evaluation for both vsomeip and CANoe are presented in Table 6.5. The table compares three scenarios: no attack, copycat attack, and de-association attack. The *Direct* column indicates the number of response messages sent directly from the regular server to the client, indicating successful communication without any attack. The *MitM* column shows the number of response messages that were redirected to the MitM attacker, indicating a successful attack.

Table 6.5.: Evaluation of Copycat and De-association Attacks

	vsomeip		CANoe	
	Direct	MitM	Direct	MitM
No Attack	458	0	498	0
Copycat	471	0	0	497
De-association	2	473	0	499

In the case of vsomeip, 458 response messages were sent without any attack, averaging 9 to 10 messages during the 10-second sending period. Unsurprisingly, the client received all messages directly from the original server.

For the copycat attack in vsomeip, a total of 471 messages were sent, but all of them were sent directly without being redirected to the attacker. This outcome is due to the implementation's restriction that only accepts new services with the same Service ID and a different IP if the time to live (TTL) of the original service has expired. In our example application, the TTL expiration only occurs after a `StopOffer` message has been sent. This behavior prevents the success of the copycat attack. While we discovered a bug in an older version of vsomeip where the basic attack was successful in overwriting a `ServiceOffer` when using UDP, the newer version integrates the TTL check for both TCP and UDP, making the attack unsuccessful. However, it is important to note that this check is not part of the specification, and different implementations may still be vulnerable to the copycat attack.

In contrast, the de-association attack in vsomeip was successful. Out of the total 475 response messages, only two were sent directly from the server to the client, while the remaining 473 messages were redirected via the attacker. The reason for missed messages

---

---

is the non-deterministic timing of the server’s responses. In rare cases, it was sent before the `StopOffer` message of the attacker.

In the case of CANoe, both the copycat and de-association attacks were successful in the request/response scenario. All messages were redirected to the attacker, indicating that the CANoe implementation favors the last `ServiceOffer`, unlike `vsomeip`. However, since CANoe is a closed-source implementation, we cannot verify this observation.

Overall, the evaluation demonstrates that the de-association attack is more effective than the copycat attack in both the `vsomeip` and CANoe implementations. The de-association attack successfully redirects the majority of response messages to the attacker, compromising the communication integrity and confidentiality. On the other hand, the copycat attack is less effective due to the implementation’s restrictions and the behavior specified by `vsomeip`.

### Evaluation of Publish/Subscribe Attack

The final attack we presented against the publish/subscribe mechanism can either make use of the copycat or de-association attack. Table 6.6 presents the results of these attack scenarios and the case without an attack for both the `vsomeip` and CANoe implementations.

Table 6.6.: Evaluation of the Attack on Publish / Subscribe

	vsomeip		CANoe	
	Direct	MITM	Direct	MITM
No Attack	556	0	500	0
Copycat	380	556	(499)	499
De-association	0 (385)	(0) 422	(500)	500

In the `vsomeip` implementation, our reference test, without any attack, produces 556 event notifications that are sent directly from the server to the client. The evaluation period was again 1000 seconds, as in the previous experiments.

For the copycat attack in `vsomeip`, the server generates event messages for a total of 556 events. Out of these 556 events, the server sends all event messages to the attacker. However, the server also generates additional event messages for some of these events (380 out of 556) and sends them directly to the client. As a result, the client receives some events twice. This behavior is due to the client’s subscription process, which occurs each time it receives an `OfferService` message from the server. Since the server regularly sends

---

`OfferService` messages, the client subscribes multiple times, leading to duplicated event messages.

In the case of the de-association attack combined with the copycat attack in `vsomeip`, the attack does not work as intended due to an implementation error. The clients ignore the first `OfferService` message after receiving a `StopOffer` message. To address this issue, we modified the attack by sending two `OfferService` messages after the `StopOffer` message. , With this modification, the attack becomes effective. The server does not remove the client from the notification list and thus sends 385 event messages directly to the client and 422 event messages to the attacker. The client, however, only accepts messages from the attacker since the client has already received the `StopOffer` message and rejects messages from the server. The total number of received event messages at the client (422) is lower than in the previous cases (556) due to the additional steps and time required for the attack to establish the MitM position.

In the CANoe implementation of SOME/IP, both the copycat and de-association attacks are effective in the publish/subscribe scenario. Unlike in `vsomeip`, the CANoe implementation server continues to send messages to the client even after receiving a `StopSubscribeEventgroup` message from the attacker. However, the client ignores these messages after receiving the `StopOffer` message, preventing any further communication between the server and the client.



Figure 6.8.: CANoe Setup

---

### 6.3.6. Security Extensions

Attacks presented on SOME/IP-SD at the application layer are still possible even when security mechanisms are deployed at lower layers. An attacker who has compromised a single ECU, not necessarily a service-offering server, can authenticate as a legitimate member of the network and perform MitM attacks.

In Section 6.3.2, we discussed related work on securing SOME/IP with an IDS [76], TLS [108], TESLA [108], and digital signatures with asymmetric cryptography [84]. We have shown in Section 6.2 that a combination of TLS and digital signatures with asymmetric cryptography can meet general automotive requirements regarding performance and have a zero false positive rate to detect potential attacks. These approaches can prevent an outsider from performing MitM attacks. However, in our attacker model, we also consider an insider attacker who can authenticate and execute MitM attacks. Additionally, TLS and the approach proposed in [84] cannot be used for SD broadcast communication.

In the following sections, we propose two security extensions for SOME/IP to prevent or at least mitigate MitM attacks on SOME/IP-SD by insiders. These extensions restrict the keys to the services that an ECU is allowed to offer and/or use, thus limiting the impact of an attacker who has compromised an ECU and obtained the keys.

#### **Secure SOME/IP Service Discovery and Session Establishment using Restricted Certificates and Digital Signatures (SESO-RC)**

In SESO-RC, each ECU is equipped with a private secret key  $sk_{ECU}$  and a corresponding certificate  $Cert_{ECU}$ , which includes the public key  $pk_{ECU}$ . A trusted certification authority (CA) issues these Certificates, and every ECU stores the public key of the CA to verify the correctness of other ECUs' certificates. Each certificate contains additional information about the services that the ECU is allowed to offer, stop, request, or subscribe to. For simplicity, we assume that all trusted certificates are preconfigured on all ECUs, and the hash of a certificate is used as a unique identifier  $ID_{ECU}$  for an ECU. Optionally, an ECU can send the entire certificate during the handshake, and receivers verify this using the locally stored trusted root CA certificate. This second approach, however, affects the performance of the protocol. The preconfigured certificates could be deployed during production and updated via firmware updates, although the secure update process is outside the scope of this paper.

A server sending an OfferService message attaches an ephemeral-static Diffie-Hellman (DH) key exchange and signs the whole message with its private key. The symmetric session key established with DH is subsequently used to calculate a MAC of all following SOME/IP messages, protecting the authenticity and integrity. A timestamp ensures the freshness of messages if ECUs are loosely time-synchronized, or a counter could be used instead.

Figure 6.9 illustrates the SESO-RC approach for request/response communication. A certificate  $Cert_S$  is stored on the server  $S$  with an access control list of services the server is allowed to offer, stop, or consume.  $Cert_S$  further contains the server's public key  $pk_S$ . It also stores the corresponding private key  $sk_S$  and the public keys of all clients. A client stores a certificate  $Cert_C$  containing the public key  $pk_C$  and a list of services the client can request. It also stores the corresponding private key  $sk_C$  and the certificates of all servers, including their respective public keys.

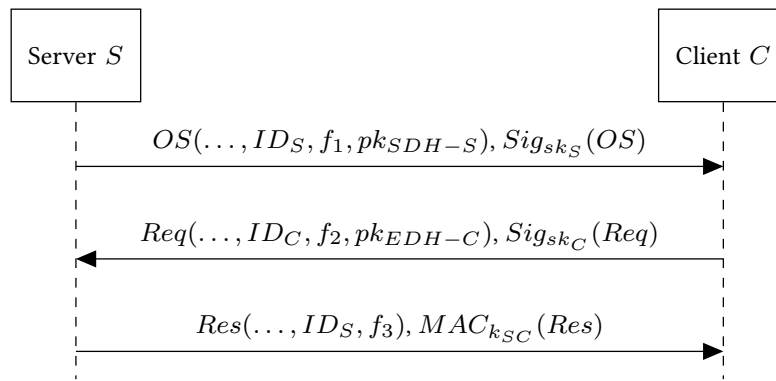


Figure 6.9.: SESO-RC with Request/Response

In the SESO-RC approach, the server broadcasts a service offer message  $OS$  that includes the usual payload, such as the service ID, instance ID, identifier  $ID_S$ , a freshness value  $f_1$ , and a DH public key  $pk_{SDH-S}$  (used by all clients later in the generation of the shared symmetric keys). This message is signed with the server's private key  $sk_S$ .

A client that wants to request a service verifies the validity of the signature and checks the freshness value  $f_1$ . If the verification is successful, the client generates an ephemeral DH key pair and includes the public part  $pk_{EDH-C}$ , a freshness value  $f_2$  that depends on  $f_1$ , and  $ID_C$  in the service request message  $Req$ . This message is signed with the client's private key  $sk_C$ .

Both the client and the server can now calculate the shared symmetric key  $k_{SC}$  using their private DH keys and the received public key. Every other client also generates its own ephemeral DH key pair and uses it with the server's key ( $pk_{SDH-S}$ ). This mechanism ensures an individual session symmetric key for every connection. Finally, the server sends the response message  $Res$ , which includes the requested service data,  $ID_S$ , and a freshness value  $f_3$ . Since the client and the server now possess the shared symmetric key  $k_{SC}$ , this and further messages are secured with a MAC.

This approach ensures that only authorized ECUs can participate in the service discovery and session establishment process, as they possess the necessary private keys and valid certificates. By restricting the keys to authorized services, the impact of an attacker who has compromised an ECU is limited to the services allowed by its certificate.

Figure 6.10 illustrates the application of SESO-RC for publish/subscribe communication in SOME/IP. In this scenario, the server broadcasts events to all clients  $C_i$  that are subscribed to the corresponding service. Since multiple clients are involved, a group key is required to secure the communication.

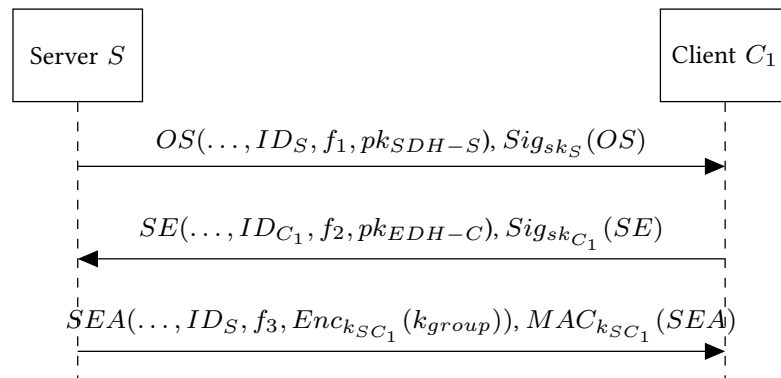


Figure 6.10.: SESO-RC with Publish/Subscribe

The initial steps of the service offer  $OS$  and subscribe eventgroup  $SE$  messages are similar to the request/response scenario. Each client  $C_i$  has established an individual symmetric key  $k_{SC_i}$  with the server.

The server generates a group key  $k_{group}$ , which is individually encrypted for each client using their respective symmetric keys  $k_{SC_i}$ , to enable secure group communication. The encrypted group key is then distributed to the clients in the subscribe event group acknowledgment message  $SEA$ . The encryption process includes a freshness value to prevent

---

replay attacks. Additionally, a MAC is calculated using  $k_{SC_i}$  and appended to the encrypted group key to ensure integrity. It is important to note that, for simplicity, we use  $k_{SC_i}$  for both encryption and MAC in this illustration. In practice, using distinct keys for each purpose is recommended, such as deriving distinct keys from  $k_{SC_i}$ .

By using SESO-RC with publish/subscribe, the communication between the server and the subscribed clients is secured with individual symmetric keys and a group key. This ensures that only authorized clients can receive and decrypt the events sent by the server. The use of encryption, freshness values, and MACs provides confidentiality, data integrity, and protection against replay attacks in the publish/subscribe scenario.

### **Secure SOME/IP Service Discovery and Session Establishment using an Authorization Server (SESO-AS)**

The SESO-AS approach utilizes very efficient symmetric cryptographic functions to reduce delays compared to time-consuming asymmetric approaches. The drawback of this variant is that it requires the authorization server (AS) to be further introduced into the E/E architecture. In the SESO-AS approach, each ECU is assigned a unique identifier  $ID_{ECU}$  and a symmetric key  $k_{ECU}$ . These values are preconfigured and stored both on the ECU and the AS. Additionally, the AS maintains information about the services that each ECU is authorized to offer, stop, request, or subscribe to. A mechanism is deployed to ensure the freshness of messages, either through loose time synchronization among all ECUs and the AS or by using a counter value.

The SESO-AS approach relies on the secure operation of the authorization server, since it acts as a trusted authority that controls and enforces the authorization of services, providing an additional layer of protection against unauthorized access and malicious activities. Since the AS is a single point of failure, we assume it is specially protected. Possible protection mechanisms in place may be a hardware security module for a secure execution environment for cryptographic functions or integrity protection of the platform like a secure boot process.

The service discovery process remains similar to the standard SOME/IP-SD. However, now the ECUs have the additional constraint of only being able to offer, request, or subscribe to services authorized by the AS. This ensures that compromised ECUs can only offer and use services within the authorized scope, mitigating the impact of an attacker.

Figure 6.11 illustrates the SESO-AS process for the OfferService. When a server wants to send a service offer message  $OS$ , it includes the usual payload (denoted with ...), its

own identifier  $ID_S$ , a freshness value  $f_0$ , the encrypted session key  $k_{se}$ , and a MAC to the AS. The AS verifies the MAC, freshness value  $f_0$ , and the authorization of the server to offer the service. If the verification is successful, the AS derives shared symmetric keys  $k_{SC_i}$  from  $k_{se}$  for both the server and the authorized clients  $C_i$  who can use (request or subscribe to) the service. These keys are individually encrypted for each client  $C_i$  using their respective keys  $k_{C_i}$ .

Next, the AS broadcasts the service offer message  $OS$  with the usual payload and a new freshness value  $f_1$ . In addition, the AS appends individual containers  $E_i$  to the message for each legitimate client  $C_i$ . For example, container  $E_1$  contains the identifier  $ID_{C_1}$ , the encrypted symmetric key  $Enc_{k_{C_1}}(k_{SC_1})$ , and a MAC  $MAC_{k_{C_1}}(OS, E_1)$ . When client  $C_1$  receives the message, it verifies the MAC, decrypts the symmetric key  $k_{SC_1}$  using its own key  $k_{C_1}$ , and uses  $k_{SC_1}$  to secure subsequent messages using a MAC.

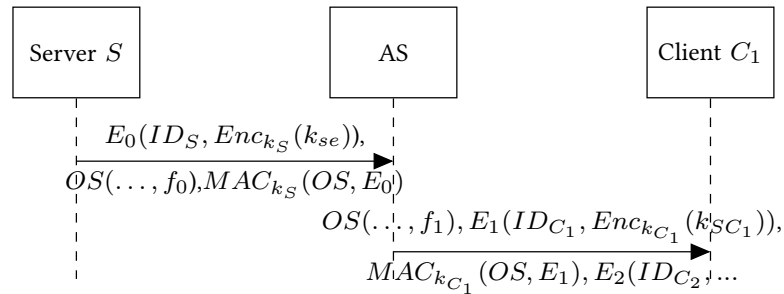


Figure 6.11.: Offer Service with SESO-AS

### Security Analysis and Verification

We model the key exchange extending SOME/IP-SD in Tamarin and use its prover to verify the security of our proposed protection mechanisms. These models are appended in Section A.2.2 and A.2.3. In contrast to our analysis of SOME/IP, we do not assume underlying security protocols but allow the attacker full control over the network. We abstract the freshness values  $f_n$  with random nonces.

**SESO-RC** For SESO-RC, we verify strong authentication of session keys as well as forward-secrecy of session keys with the following two security properties (cf. Section 6.3.3):



---

**Definition 2** (Injective-agreement on session keys). *For an honest server  $S$  and honest client  $C$ , it holds that whenever  $S$  establishes a session, apparently with  $C$ , then  $C$  has previously established a session, apparently with  $S$ , and  $C$  and  $S$  agree on all parameters of the session, especially the session key. Moreover, each session established by  $S$  corresponds to a unique session established by  $C$ .*

Injective agreement not only implies aliveness (cf. Section 7.4) but also protection against replay and MitM attacks, given that no ECU is authorized to offer and consume a service at the same time.

**Definition 3** (Forward-secrecy of session keys). *For an honest server  $S$  and honest client  $C$  and a session key  $k$  that either of the two parties believes to be a session key shared with the other party, the attacker cannot learn  $k$  even if she compromises the private keys of  $S$  or  $C$  later.*

**SESO-AS** We assume for the formal verification, the authorization server  $AS$  will only allow key exchanges for a specific service between authorized servers and clients. Therefore, we do not represent different authorization groups within our model. We require *injective agreement* on the shared keys between a client and a server  $k_{SC_n}$  as well as *secrecy* of the session keys  $k_{se}$  and  $k_{SC_n}$ . An elementary difference in the security properties between SESO-AS and SESO-RC is that SESO-RC additionally provides *perfect forward-secrecy* for session keys because SESO-AS encrypts session keys always with the same key while SESO-RC derives a key from DH key exchange. Both properties are verified in Tamarin.

**Definition 4** (Injective agreement on shared keys). *For an honest server  $S$ , honest client  $C$ , and honest authorization server  $AS$ , it holds that whenever  $C$  receives an `OfferService` message, apparently from  $S$ , then  $S$  has sent an `OfferService` message before and the shared symmetric key  $k_{SC}$  received by  $C$  has been correctly derived from the session key  $k_{se}$ . Moreover, each received event by  $C$  corresponds to a unique send event by  $S$ .*

**Definition 5** (Secrecy of shared keys). *For an honest server  $S$ , honest authorization server  $AS$ , and every session key  $k_{se}$  generated by  $S$ , the attacker cannot learn  $k_{se}$ .*

*In addition, for an honest server  $S$ , honest client  $C$ , and honest authorization server  $AS$ , it holds that whenever  $C$  receives an `OfferService` message with a shared symmetric key  $k_{SC}$ , apparently shared with  $S$ , the attacker cannot learn  $k_{SC}$ .*

---

## Performance Evaluation

In this section, we present the implementation of both solutions as well as evaluate the performance overhead introduced by SESO-RC and SESO-AS. We compare the performance of these approaches with "Secure SOME/IP" [84], the solution [108] is excluded from this comparison since it does not effectively protect the service discovery messages.

To begin with, we discuss the communication overhead resulting from the increased message size in the three approaches. We then analyze the impact of the additional data added to the SOME/IP messages to support the security extensions, including certificates, signatures, session keys, and other security-related information.

Next, we evaluate the computational overhead of the three approaches during the communication process. This includes the time required for cryptographic operations such as signature generation and verification, encryption and decryption, and key derivation. By measuring the execution time of these operations in each approach, we can assess the computational impact of the security extensions.

Finally, we summarize the performance evaluation of SESO-RC, SESO-AS, and "Secure SOME/IP" by considering both the communication overhead and computational overhead. We provide a comparative analysis of the three approaches, highlighting their strengths and weaknesses in terms of performance. This evaluation will help assess the feasibility and suitability of each approach in different scenarios and provide insights into the trade-offs between the security and performance of the different solutions.

## Message Overhead

Regarding message overhead, SESO-RC and SESO-AS introduce additional data to the SOME/IP messages to support the security extensions.

In *SESO-RC*, each message includes a freshness value  $f$  of 4 bytes. The `OfferService` message of the server includes the 32-byte public key for elliptic curve Diffie-Hellman (ECDH) key exchange (X25519) and the 32-byte SHA-256 hash value of the certificate. A 64-byte Ed25519 elliptic-curve signature also accompanies the message. Figure 6.12 illustrates the complete message format of the `OfferService` message in *SESO-RC*. The first message (a `Request` or `SubscribeEventgroup`) sent by the client has an identical format. The subsequent messages of the client and server include a 32-byte HMAC-SHA-512 (truncated to 256 bits) for integrity protection and a 4-byte freshness value.

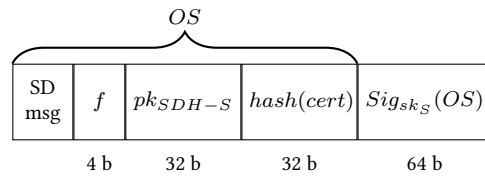


Figure 6.12.: Message Format of Offer Service with SESO-RC

In *SESO-AS*, each message includes a freshness value  $f$  of 4 bytes, along with a set of encrypted keys for each possible client in the `OfferService` message. For encryption and authentication, an Authenticated Encryption with Associated Data (AEAD) construction (ChaCha20-Poly1305) is used since we found this to be a performant cipher in Section 6.2.4. The required nonce for the AEAD construction is the freshness value of the message. An entry in the key set consists of a 1-byte identifier of the client ( $ID_{C_i}$ ), the 32-byte encrypted key  $k_{SC_i}$ , and a 16-byte AEAD authentication tag. The authentication tag includes the `SOME/IP OfferService` message as associated data. The complete message format is depicted in Figure 6.13. Similar to *SESO-RC*, subsequent messages in *SESO-AS* include a 32-byte HMAC-SHA-512 (truncated to 256 bits) and a 4-byte freshness value.

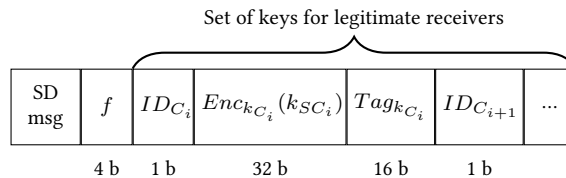


Figure 6.13.: Message Format of Offer Service with SESO-AS

A regular `SOME/IP` message has an overhead of 36 bytes in both *SESO-RC* and *SESO-AS*. However, the key exchange handshake differs depending on the number of clients. Table 6.7 provides a comparison of the different approaches for varying numbers of clients, indicating the number of additional messages (*msg*) and the total number of additional bytes.

*SESO-AS* introduces a message overhead of 53 bytes in the service offer messages sent to the AS, along with an additional `OfferService` message of  $4 + 49 \cdot c$  bytes, where  $c$  is the number of legitimate clients. This additional message is broadcast to every client in the network.

*SESO-RC* does not introduce additional messages for the key exchange since the service

Table 6.7.: Overhead of Cryptographic Key Exchange

		Number of Clients				
		1	2	4	8	16
SESO-AS	msg	1	1	1	1	1
	bytes	106	155	253	449	841
SESO-RC	msg	0	0	0	0	0
	bytes	264	396	660	1188	2244
"Secure SOME/IP" [84]	msg	0	2	4	8	16
	bytes	180	360	720	1440	2880

offer message is directly broadcast to all potential clients. Both the service offer and the first request have an overhead of 132 bytes. An additional 132 bytes of overhead is added for each additional client requesting the service.

For "Secure SOME/IP" [84], which uses a unicast handshake between each client and the server, we replaced RSA with Ed25519 EC signatures for a fair comparison. The client part of the handshake includes 36 bytes of the certificate hash along with a freshness value. The server adds 144 bytes, which contain the asymmetrically encrypted and signed group key. Both messages are sent for each additional client.

When comparing the three approaches regarding message overhead, SESO-RC introduces the fewest additional messages, while Secure SOME/IP introduces the most messages. This is the case due to the fact that Secure SOME/IP loses the ability to broadcast service discovery messages. SESO-AS has the most negligible overhead in message size, particularly when used with multiple clients.

**CPU Overhead** To evaluate the CPU overhead introduced by the security extensions, we implemented SESO-RC, SESO-AS, and Secure SOME/IP on ARMv6 boards (ARM1176JZF-S at 700MHz) connected via a 100 MBit/s switch. We utilized the libsodium<sup>4</sup> cryptography library for the implementation.

The latency evaluation was performed by measuring the additional number of CPU instructions using the `perf_events` feature of the Linux Kernel. The measurements did not include network transfer time or the interpretation of SOME/IP messages. For SESO-RC and Secure SOME/IP, we evaluated both the client and server sides, while we additionally

<sup>4</sup><https://github.com/jedisct1/libsodium>

evaluated the AS side for SESO-AS. We conducted 1000 measurements for different numbers of clients and calculated the arithmetic mean and mean absolute deviation (MAD) to assess the measurement error. The results are presented in Figure 6.14.



Figure 6.14.: Additional CPU Instructions for Cryptography

SESO-AS clients and servers exhibited a very low overhead of only 7,032 instructions (MAD: 13.1) and 9,182 instructions (MAD: 225.2), respectively, irrespective of the number of clients. The authorization server required a maximum of 153,937 instructions (MAD: 3,615.6) when encrypting a broadcast message for 16 clients. As expected, SESO-AS outperformed SESO-RC, which required an additional 4,134,880 instructions (MAD: 22,059.0) for the server and 4,134,550 instructions (MAD: 21,707.0) for the client.

Secure SOME/IP was faster (2,939,572 instructions, MAD: 15,371.1) compared to SESO-RC on the client side. The signature verification of every client in SESO-RC resulted in significantly higher CPU usage for the server compared to Secure SOME/IP, which did not

---

verify the clients. Both SESO-RC and Secure SOME/IP experienced an equal increase in CPU usage with a larger number of clients, while SESO-AS exhibited a moderate increase.

It is important to note that Secure SOME/IP requires additional time for the unicast communication of service offers, which adds to the overall delay. In an established system, previously exchanged symmetric keys could be utilized to validate client requests for new keys, significantly improving server performance.

**Summary** SESO-AS outperforms the other approaches in terms of latency and message overhead. However, it introduces a single point of failure regarding security and safety. Moreover, integrating new devices into the E/E system becomes more complex since the permission list of the authorization server needs to be updated, and keys need to be exchanged in advance.

SESO-RC addresses these issues by employing certificates that can be easily distributed, and the distributed authorization can be seamlessly integrated into the SOME/IP protocol. SESO-AS requires a minor protocol change to send all service discovery messages to the authorization server. Both approaches support broadcast service discovery, which Secure SOME/IP does not.

With sufficient processing power, SESO-RC is the preferred option as it avoids a single point of failure, provides forward secrecy, and simplifies the distribution of keys and permissions through certificates. Hardware-based cryptography accelerators may enhance performance for low-powered devices but at an additional cost. Alternatively, SESO-AS can be employed with ECUs with limited computational power. However, ensuring the security of the authorization server becomes crucial for the overall network security.

All three approaches effectively protect against the identified MitM attacks. Moreover, SESO-RC and SESO-AS provide the ability to establish individual secure channels between clients and servers, as well as secure broadcast channels for subscriptions.

## 6.4. Summary on Automotive Ethernet Security

In conclusion, the proposed communication strategy, assuming the security assumptions of TLS hold, provides guarantees of authenticity through the handshake, as well as confidentiality and integrity through channel encryption and authentication codes. Through our investigation of real-world performance requirements for TLS in in-vehicle networks,

---

we have found that these requirements can generally be met using typical automotive hardware without the need for a dedicated encryption hardware module. Regarding performance, we recommend using ChaCha20 + Poly1305, as it satisfies most of the requirements. However, we have also explored more experimental algorithms, such as HC-128 and Rabbit, which promise improving throughput and latency.

Additionally, we have presented a trust architecture based on a public-key infrastructure, enabling the exchange of ECUs between vehicles and facilitating interactions between vehicles, backends, and diagnostic devices. We have also discussed mechanisms to address the limitations of automotive systems, such as the lack of a trusted time source and the limited time available to establish connections with key exchanges.

However, it is important to note that this does not cover an essential aspect of Automotive Ethernet communication: service-oriented communication in vehicles. After conducting a comprehensive analysis of SOME/IP using Tamarin, we have identified three distinct types of MitM attacks that can be executed on SOME/IP, even when link-layer security mechanisms are in place. These attacks include the copycat attack and the de-association attack on the service offer, which allow an attacker to gain a MitM position in request/response communication. The third attack targets publish/subscribe communication and can be executed by combining the first two attacks.

Our implementation and evaluation of these attacks on two real-world SOME/IP libraries, `vsomeip` and `CANoe`, have demonstrated their effectiveness. In the case of `vsomeip`, the de-association attack proves to be highly successful, as it redirects almost all messages through the MitM attacker. When attacking publish/subscribe communication in combination with the copycat attack using `vsomeip`, over two-thirds of the messages are routed through the attacker. By incorporating the adapted de-association attack, all messages received by the client are sent through the attacker, albeit with some loss of event messages from the server. On the other hand, `CANoe` is susceptible to both attacks in request/response and publish/subscribe scenarios, with the regular server continuing to send messages to the client in the publish/subscribe case.

We propose two security extensions to counteract these attacks: Secure SOME/IP service discovery and session establishment using restricted certificates and digital signatures (SESO-RC) and Secure SOME/IP service discovery and session establishment using an authorization server (SESO-AS). We have conducted a Tamarin analysis to evaluate the security of these extensions, and our implementation supports their practical feasibility.

SESO-RC is particularly suitable for scenarios where ECUs have sufficient resources for asymmetric cryptography. Notably, SESO-RC allows for the continued use of broadcast

---

communication for service discovery, distinguishing it from related work. Alternatively, SESO-AS offers a viable solution for resource-constrained ECUs by relying solely on efficient symmetric cryptography. However, it introduces an additional component in the form of the authorization server (AS).

By leveraging these security extensions, automotive systems employing SOME/IP can be better protected against the identified MitM attacks. The combination of formal analysis, practical evaluation, and the proposal of security extensions provides a comprehensive approach to enhancing the security of SOME/IP communication.

Regarding the attack path feasibility for an *Advanced Remote Attacker* causing unintended driving behavior after taking over the TCU in our model, SESO-AS or SESO-RC decreases the probability of attacks against service-oriented communication. ECUs can no longer send legitimate messages of services they do not originally use unless the attacker breaks the cryptographic key or finds an exploit for an ECU with a direct connection to a safety-critical system. This increases the attack feasibility to a very low rating.



---

## 7. Improved Security and Privacy for Charging Systems

---

There are already over 16 million fully battery-powered or plug-in hybrid EVs worldwide [61], and their penetration in suburban and rural areas is expected in the near future [180]. Many private households install CSs to recharge EV batteries at home.

These vehicles introduced a new communication channel for the charging process to exchange charging schedules and parameters as well as the authentication of charging processes via PnC. Furthermore, use cases like bidirectional charging emerge, allowing grid operators to use the batteries to store power in the vehicle battery and use this energy later when not much power is produced. The risk assessment in Chapter 4 indicates a high risk due to the catastrophic effects of a blackout but also privacy risks.

For these reasons, we first discuss possible attacks against the charging infrastructure in this chapter, subsequently resulting in attacks against the power grid, which is based on our paper [212], where I contributed the attacks on EVs and CSs from a theoretical and practical perspective, including the implementation of a MitM box to attack communication between vehicle and charging stations. Additionally, we investigate the privacy impacts of the charging communication and suggest an extension for ISO 15118 to prevent the leakage of personal data by using DAA. This part of the chapter is based on [204], where I contributed the analysis of the different charging protocols for personal data, designed the privacy-preserving protocol, and analyzed its security.

These publications lead to multiple impacts. First, we had discussions with Electrify America, and a German OEM regarding the security and privacy of the current charging infrastructure that finally led to multiple industry projects with this OEM. Furthermore, we participated in the CharIN Cybersecurity Workshop as well as being invited as an expert to a workshop of the Nationale Leitstelle Ladeinfrastruktur by the BMBV, to the Ladenburger Diskurs, and the Dialogkreis VDA-Datenschutzaufsichtsbehörden.

---

This chapter is structured as follows: First, we provide an introduction to the fundamentals of DAA, later used in the privacy-preserving solution. Following this, we investigate related work concerning grid and V2G attacks. In the subsequent sections, we evaluate potential attack vectors and conduct a comprehensive security and privacy analysis of EV charging. Additionally, we present various attacks on charging communication and offer practical assessments of these attacks against real-world CS. For each attack, we provide an assessment of its practicality and offer mitigation strategies to address the identified vulnerabilities. To conclude, we present our privacy enhancement approach for the charging process, encompassing a requirement analysis and a detailed security discussion.

## 7.1. Basics on Direct Anonymous Attestation (DAA)

Direct Anonymous Attestation (DAA) plays a critical role in our proposed privacy-enhanced charging solution. We provide a brief introduction to DAA in this section.

DAA is a remote authentication and attestation protocol used in a trusted computer system to preserve the privacy of the system's user. The Trusted Computing Group (TCG) has adopted this protocol for use with the TPM. Attestation allows a third party to verify that the software on a system has not been altered and is in a trustworthy state.

Before adopting DAA, the TCG specified a Privacy-Certified Attestation System (PCAS) with Version 1.2 of the TPM [176], which is also standardized in ISO/IEC 11889 [85]. In Privacy-CA solution (PCAS), a Privacy-CA (Certification Authority) acts as a trusted third party that needs to be contacted before each anonymous authentication or attestation. Instead of using the embedded RSA key pair (Endorsement Key (EK)) for authentication, PCAS uses a new RSA key pair called Attestation Identity Key (AIK) for each authentication. An EK is a unique asymmetric key pair embedded in the TPM during manufacturing. It's used to endorse or sign other keys and certificates within the TPM, contributing to the platform's overall security. The TPM sends the public AIK, signed by the EK, to the Privacy-CA, which then verifies its validity and issues a certificate for the AIK. However, PCAS has several drawbacks, such as requiring high availability and security for the Privacy-CA and potential privacy violations if the Privacy-CA colludes with the verifier.

Based on RSA, the DAA protocol initially described in [23]. It involves Hosts, an Issuer, and multiple Verifiers. Each Host is equipped with an integrated TPM security chip. The Issuer generates and maintains a group of trusted systems and decides which Hosts and

---

---

associated TPMs can become members of this group. Verifiers can check whether a Host is a member of the group.

Our concept employs the DAA protocol presented in [28], which overcomes the weaknesses of earlier DAA schemes. We briefly introduce the basic functions of DAA:

1. **Setup:** The Issuer  $I$  creates a key pair  $sk_I, pk_I$  during the setup phase. This process is represented as  $setup(I) = sk_I, pk_I$ .
2. **Join:** During the join protocol, a Host  $H$  joins the group of an Issuer. Authenticated by the Issuer, the TPM and Host obtain a credential  $sk_H$  enabling the Host to create group signatures. To ensure security, part of the secret to create signatures is stored on the TPM during the join process, preventing a compromised Host from leaking keys. As a result, the Host can only sign in collaboration with the TPM. This process is represented as  $join(I) = sk_H$ .
3. **SignDaa:** The Host can sign an arbitrary message  $m$  with respect to a basename  $bsn$ . This process is represented as  $signDaa(m, bsn, sk_H)$ .
4. **Verify:** Any Verifier can then verify the signature using the public key  $pk_I$  from the Issuer. This process is represented as  $verify(m, pk_I)$ .
5. **Link:** Additionally, Verifiers have the ability to link different signed messages if, and only if, the basenames  $bsn$  of two messages  $m$  and  $m'$  signed by the same TPM are equal. This process is represented as  $link(m, bsn, m')$ .

These procedures ensure anonymity, making it impossible to distinguish whether the same or different honest Hosts created two signatures. Moreover, no adversary can create valid signatures, and no TPM can create a signature linked to another TPM.

## 7.2. Related Work of the Security of EV Charging

**Attacks on Power Grids:** In [39], researchers devised attacks that impacted grid frequency through coordinated modulation of power consumption in a botnet of compromised computers. These attacks had the potential to trigger unstable states within the grid, requiring a substantial number of infections ranging from 2.5 to 9.8 million. The study primarily focused on the grid's response to production-consumption imbalances, including aspects of self-regulation and primary control.

---

In [170], an investigation delved into a botnet attack involving high-wattage IoT devices. This research analyzed the consequences of such attacks, including frequency instability, line failures leading to cascading failures, and escalating operating costs. The primary emphasis was on dynamically simulating power flows within the transmission network.

Kern et al. proposed a framework in [99] for simulating and analyzing the impact of e-mobility-based attacks on grid resilience. This framework was applied in various case studies to evaluate the effects of e-mobility-based attacks on grid resilience.

In contrast to these studies, our research in [212] focuses on manipulations at the lowest grid level in our simulations. Here, grid frequencies remain unaffected, and the threat lies in overload situations that can result from an attack. Our approach fundamentally differs from the works as mentioned earlier as it does not require dynamic consumer and producer models. Instead, we concentrate on modeling a local grid with a realistic structure, incorporating innovations such as photovoltaics, heat pumps, and EVs while ensuring the integration of consistent time series data on household consumption and production.

While previous research has examined the vulnerability of smart grids to attacks against IT-based control systems, such as in [116, 201, 2], these studies did not consider local grids or recent V2G communication protocols. Notably, a real-world attack on a DSO is documented in [209], highlighting practical vulnerabilities. Additionally, in [196], control over a power plant was compromised due to an attack on the communication link.

**Attacks on V2G Components:** Regarding the real-world security of cyber-physical systems in the context of V2G, practical attacks on chargers and EVs from multiple manufacturers have been demonstrated. In [40], researchers showcased the installation of manipulated firmware on a charger via a USB port alongside an intriguing example involving a high-wattage device (a waffle iron) masquerading as a charging vehicle.

In [32], authors employed reverse-engineering techniques to manipulate the firmware updates provided online by a CSO, gaining remote control over its charging points. Furthermore, penetration tests conducted on charge points from various brands, as reported in [174], uncovered additional vulnerabilities, enabling adversaries to exert complete control.

In [105], a reported DoS attack on PLC systems was capable of remotely halting EV charging. This attack prevented EV charging remotely. Moreover, the possibility of eavesdropping on plaintext ISO 15118 messages via PLC was demonstrated in [9] and [48].

The associated testbed in [48] empowered attackers to read and manipulate the entire communication stack, allowing for the injection of crafted V2G messages. These works complement our research, contributing to a broader understanding of V2G security challenges.

### 7.3. Security and Privacy Analysis of EV Charging Protocols

In this section, we analyze the privacy and security of EV charging protocols. Out of a large number of proposed communication protocols in the context of EV charging [149, 136, 72], we concentrate on the protocols ISO 15118, OCPP, and OICP as introduced in Section 2.3. The reference architecture introduced in Section 2.3.1 is displayed again in Figure 7.1.

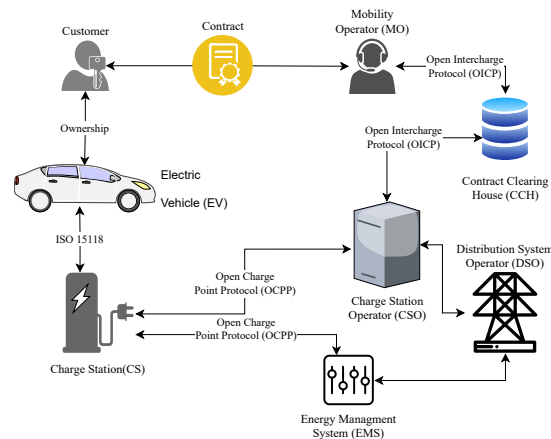


Figure 7.1.: Example Network Architecture of Modern Electric Vehicle

For the purpose of the privacy and security analysis, we first describe possible attack vectors one of the attackers described in Section 3.3 might attack. Second, we review the security issues of the given charging protocols, and finally, we analyze the charging process transfer for personal information.

---

### 7.3.1. Attack Vectors

An adversary, as described in Section 3.3, can use the following attack vectors to gain unauthorized control over the V2G system:

- Communications: ISO 15118 stack over PLC (charging cable), E/E communications (CAN, CAN FD, or Automotive Ethernet), and OCPP-based or proprietary connections (usually via cellular networks) as well as OICP-based communication between servers, and signals from the DSO;
- Local systems: EV (EVCC, E/E system), and CS (SECC) for example via debugging or maintenance ports;
- Backend systems: DSO, CSO (remote access to a set of charge points), CCH, MO, or OEM (remote access to a set of EVs);
- EV charging processes: EV authentication, charging authorization, billing information, session handling, parameter (re)negotiation, target setting, system management, and personal data.

Our *Local Attacker* can perform an attack against communication since he has access to every communication channel, including cellular communication. Our *Advanced Remote Attacker* is capable of accessing local systems. The third attacker (*Internal Attacker*) has access to a backend system and thus can manipulate messages or gather information at a CSO or CCH server.

### 7.3.2. Security Analysis of EV Charging

Building upon the previously introduced attack vectors, we have examined potential attacks that could destabilize the power grid, a severe concern of our threat assessment (cf. Chapter 4). In our security analysis, we assume that the DSO possesses the capability to monitor grid conditions, particularly EV charging activities. In the event of a critical increase, the DSO is able to issue a signal to curtail consumption through a hierarchical chain of command: CSO→CS→EV, targeting a selected subset of CSOs. A delay affects all these communication paths; thus, a signal does not trigger an action in the charging process immediately. To execute a successful attack, the adversary must either achieve a grid overload before charging is throttled or counteract this action as part of their attack strategy. For instance, forcing EV charging to pause initially can thwart the DSO's early intervention.

---

---

ISO 15118 incorporates security controls such as TLS, XML Security, PKI-based authentication, and private key protection [90, 88]. Conversely, OCPP 1.6 [140] is inherently vulnerable. The security framework enabling optional secure connection establishment via TLS 1.2, along with secure remote firmware updates, was introduced with OCPP 2.0 in 2018 and subsequently adapted for OCPP 1.6 [82]. Legacy systems still permit the use of insecure TLS 1.0 and 1.1. Notably, OCPP 1.6 has been identified as susceptible to MitM attacks, DoS attacks, and data tampering, with the potential to disrupt energy services and induce overloads [3, 155]. In ISO 15118-2, issues related to time synchronization, certificate validation, and optional TLS encryption have been reported [54, 10].

We verified these findings and anticipated that real-world products may exhibit vulnerabilities due to the complexity of specifications, the presence of optional security features, and ambiguous security assumptions. Below, we present a concise summary of our security analysis of ISO 15118-20 [88] and OCPP 2.0.1 [141], which has not been previously documented.

1. *Insecure Channels*: The underlying TLS channel is utilized to secure data exchange on the application layer in ISO 15118-2 and -20. However, ISO 15118-2 mandates TLS solely for PnC, while in scenarios involving external identification and private environments like households, communication occurs in plaintext [90]. Although ISO 15118-20 consistently deploys mutual TLS, certificate validation can be bypassed in private environments, permitting the installation of self-signed (root) certificates [88]. Furthermore, it allows a fallback to -2 for legacy support (see next point).
2. *Version Downgrade*: ISO 15118-20 supports mandatory mutual TLS with forward-looking cipher suites and key lengths. However, a less secure ISO 15118-2 can be negotiated during session establishment, enabling unencrypted communication for EIM and in private environments. Implementing optional security measures is at the vendors' discretion, and TLS is frequently omitted [9]. Adversaries can manipulate systems into downgrading and intercepting all transmitted data.
3. *Insecure PLC*: The integration of HPGP into ISO 15118 [91] lacks sufficient encryption and is prone to eavesdropping and MitM attacks. Consequently, attackers can effortlessly obtain the Network Membership Key used for authentication and join the network, as demonstrated in Section 7.5.2.
4. *Session Hijacking*: During ISO 15118 session resumption, presenting the `SessionID` of the previously authenticated session suffices. Since TLS is optional, this data can be eavesdropped by adversaries and employed as a bearer token for impersonating

---

the original EV. This enables the manipulation of charging sessions, renegotiation of charging profiles, or charging on the victim's account. While ISO 15118-20 recommends binding `SessionID` to mutual TLS certificates, these certificates are not part of application-layer authentication, allowing potential falsification of endpoints' entries. The Vehicle to Grid Transport Protocol (V2GTP) tunnels via TLS, but session management transpires on the application layer. The absence of channel binding may also lead to session hijacking.

5. *No End-to-End Guarantees*: ISO 15118's scope extends solely to communication between an EV and a CS. Data protection ceases once it reaches the CS. Backend systems, involved in most charging processes, are not covered. Consequently, there is no end-to-end data integrity and confidentiality assurance between the EV and the recipient, enabling chargers to manipulate backend communications.
6. *Cryptographic Key Reuse*: Neither standard adequately defines key and certificate management. Issues related to certificate and key revocation and deletion remain unaddressed. ISO 15118 permits the reuse of authentication keys for signing and encryption purposes.
7. *Non-Existent PKI Services*: ISO 15118 and OCPP rely on the V2G PKI to establish trust. Presently, a globally trusted party of this nature does not exist, making it challenging to verify identities with high confidence. Certificate profiles lack the detail required for complete syntactic and semantic validation.

These findings provide an assessment of the security landscape, shedding light on potential vulnerabilities in ISO 15118 and OCPP. Both ISO 15118 and OCPP protocol have multiple parameters with the potential to influence the EV charging behavior and thus are of particular interest to the adversary:

- *Charging Profile/Schedule*: These parameters define future time periods during which either the EV or CS actively consumes or transfers energy;
- *Tariffs*: EVs can be programmed to charge during periods of low energy prices through tariff updates obtained from backend systems;
- *State-of-Charge*: Charging cycles have a direct impact on battery longevity, and EV charging systems are designed to prevent overheating, overcharging, or fully discharging the battery;
- *User Preferences*: Users can set preferences that dictate when the EV should be charged and ready to depart, including the target state of charge;



- 
- *Control Signals from DSO/CSO*: Control signals from the DSO or CSO can instruct CSs or EVs to reduce or increase their energy consumption based on the prevailing grid conditions.

The ability to manipulate these parameters grants an adversary significant control over EV charging behavior and, consequently, exerts influence over the associated grid load.

### 7.3.3. Privacy Analysis of EV Charging

In this section, we provide a comprehensive analysis of personal data flow throughout the entire PnC charging sessions process chain. This analysis aims to identify the various actors and processes responsible for collecting and managing personal data.

The term *personal data*, as defined by the General Data Protection Regulation (GDPR) [53], encompasses information related to users of PnC services. This includes data that, either on its own or when combined with supplementary information, can reveal insights into users' private lives. Such data can be used for user profiling, tracking, or potentially exposing a user's identity. Recent studies [34] have highlighted the extent of privacy-sensitive information leakage from connected vehicles. This information could potentially disclose details about an EV owner's residence, profession, interests, or even their health.

A closer examination of PnC communication reveals additional privacy risks. From a privacy perspective, we systematically evaluate each step within the ISO 15118 protocol, as defined in [90]. This protocol, namely OCPP and OICP, plays a central role in shaping our system architecture (cf. Section 2.3.1). Two additional protocols come into play within the system architecture of the CS and CSO backend systems due to the current state of the ISO 15118-2 standard and its real-world implementation. We used the official message names from the different standards so that the names may look inconsistent.

For the analysis, we focus on a charging session with roaming, where all key stakeholders participate. We also assume that the CS is online during the communication. The charging process is divided into five essential parts:

1. Communication Setup
2. Identification, Authentication, and Authorization
3. Target Setting and Charge Scheduling
4. Charge Control and Rescheduling

---

## 5. End of the Charging Process

**Communication Setup** The EVCC of an EV initiates communication with the SECC of the connected CS (displayed in [fig:chargComSetup]). A TLS session is established, where the CS authenticates itself using its SECC Leaf Certificate. From this point onward, the EVCC communicates with the SECC using the ISO 15118 protocol. First, in the *supportedAppProtocol* request/response, EVCC and SECC negotiate the protocol version. The EVCC transmits the *SessionSetupReq* with its unique identifier *EVCCID* to the SECC in the following step. This request also establishes a session ID for the communication. The CS responds with its unique *EVSEID* and a *timestamp*. After establishing the session, the EVCC requests the list of services the CS offers using the *ServiceDiscoveryReq*. Subsequent to the initial *ServiceDiscoveryReq*, (optionally) requests for additional details using the *ServiceDetailReq*, the EVCC selects a set of offered services and transmits a *PaymentServiceSelectionReq* selecting the payment option used for the charging process. In our scenario, PnC is the chosen payment option, also named *Contract*, which results in the use of Contract Certificates. In case the EVCC does not possess a valid Contract Certificate, it now requests the installation of a new certificate using a *CertificateInstallationReq* containing its unique *OEM Provisioning Certificate* and the respective certificate chain. Otherwise, if the EVCC has a valid contract certificate, it can request its update using a *CertificateUpdateReq*. This request contains the *EMAID* as well as the current contract certificate and certificate chain. ISO 15118 does not specify the exact process, but it assumes that all roles and components described in Section 2.3.1 are involved [89, 90].

**Identification, Authentication, and Authorization** After selecting the payment method, the EV suggests payment details using the *PaymentDetailsReq*. This message includes *EMAID*, *Contract Certificate*, and the corresponding certificate chain of the EV. The SECC responds with a challenge starting the authentication phase (Figure 7.3). The EVCC signs the challenge using its private key of the Contract Certificate and transmits an *AuthorizationReq* containing the signed challenge back to the SECC requesting authorization to begin a charging process. The SECC validates both the Contract Certificate and certificate chain and uses the *EMAID* to request authorization from the backend using an *Authorize.req*. If the *EMAID* belongs to an MO that does not have a direct contract with the CSO, it uses roaming services to authorize the charging session. The CSO sends the *eRoamingAuthorizeStart* message containing *EMAID*, possibly the *EVSEID*, and *SessionIDs* to request authorization from the respective MO via the CCH. The CCH analyzes the prefix of the *EMAID* and forwards the request to the respective MO. At this point, all participants

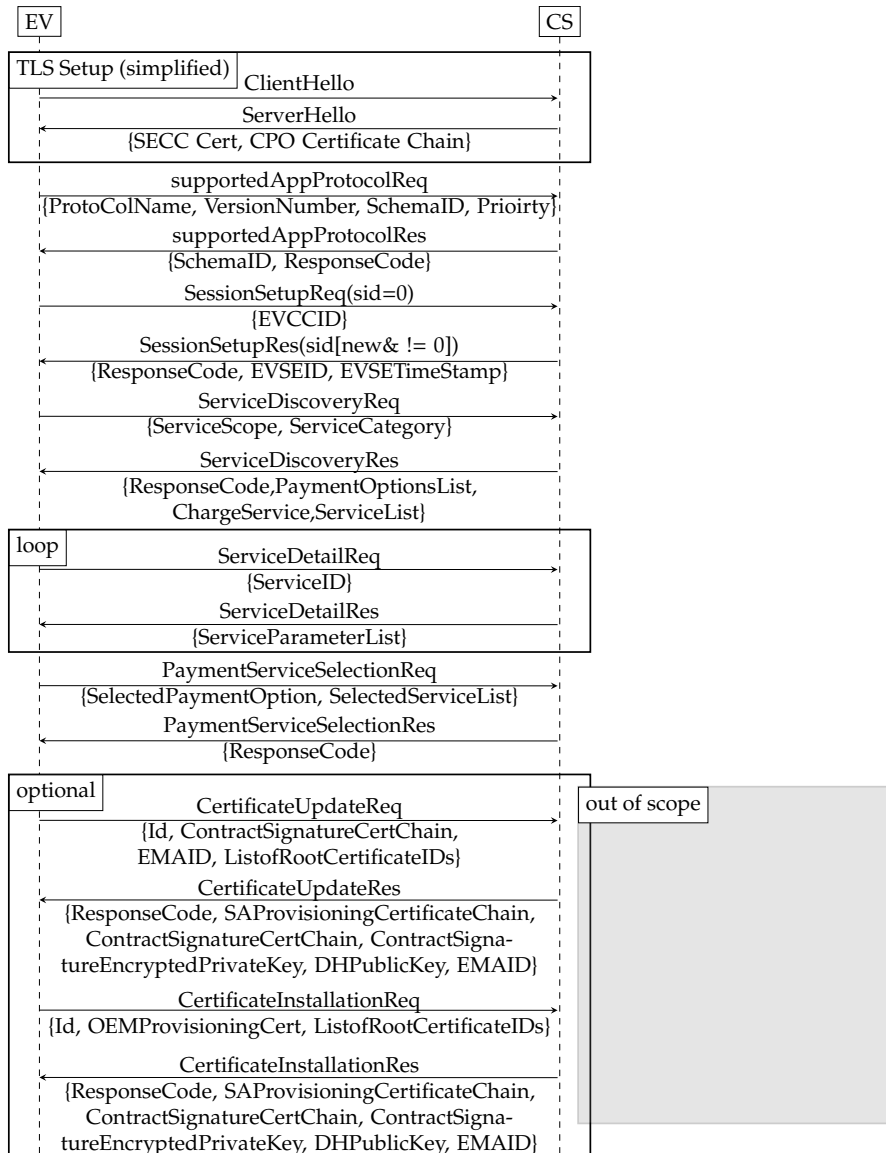


Figure 7.2.: Communication Setup

in the system received the customer's *EMAID*. The MO verifies if the *EMAID* belongs to a valid contract and sends an *eRoamingAuthorizationStart* indicating the authorization sta-

tus. The CCH forwards this to the CSO backend, which sends the *Authorize.conf* to the CS, stating whether the request was authorized and how long the authorization will be valid. According to the OCPP 1.6 specification [140], the CS can be configured to cache this authorization together with all other recent authorizations until they expire. This makes the CS an interesting target for attackers to determine which customers have charged their EVs at a specific CS. The sequence is finished with a positive *AuthorizationRes* to the EVCC.

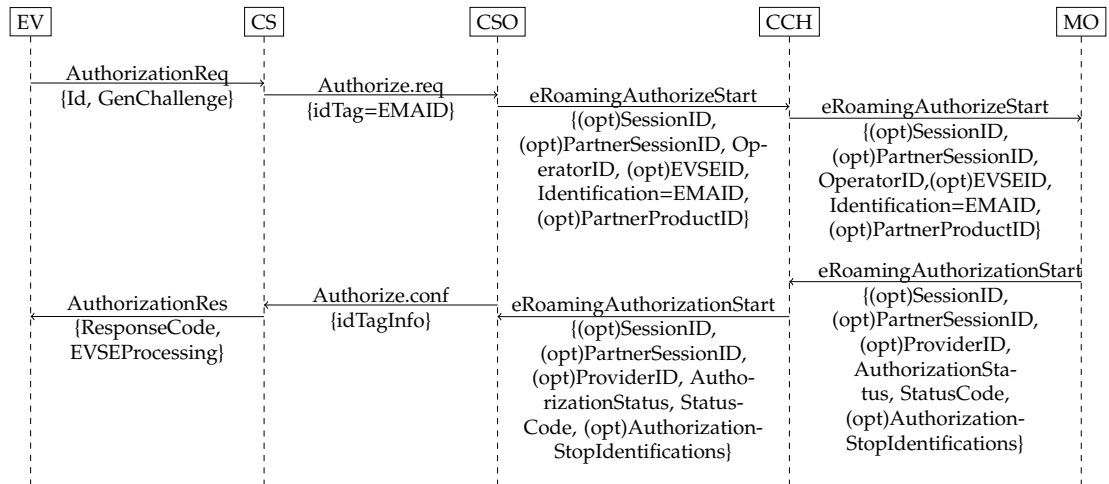


Figure 7.3.: Identification, Authentication, and Authorization

**Target Setting and Charge Scheduling** Now that the authentication is finished, the settings for the session can be negotiated (Figure 7.4). Initially, the EVCC can request the SECC to get a tariff list for the given *EMAID* and charging mode (AC or DC) from the MO using the *ChargeParamterDiscoveryReq*. Since this process between the CS and MO is not specified in any of the three protocols, we consider this request out of scope for further analysis. It should be noted, though, that the CS will probably request the tariff tables using both the *EMAID* and information on AC or DC charging so that all stakeholders routing this request might get access to this information. Once the EVCC has received the schedules, a *PowerDeliveryReq* request will be transmitted. Upon confirmation from the SECC, the CS will provide voltage to the power outlet, thereby initiating the charging process of the EV. Before responding to the request, the CS will notify the CSO backend about the start of the charging session using the *StartTransaction.req* with the respective connector ID (power outlet), the *EMAID*, the current *meter value*, a *timestamp*,

and optionally a *reservation ID*. The response to the EVCC is sent once the CS receives the confirmation from the backend.

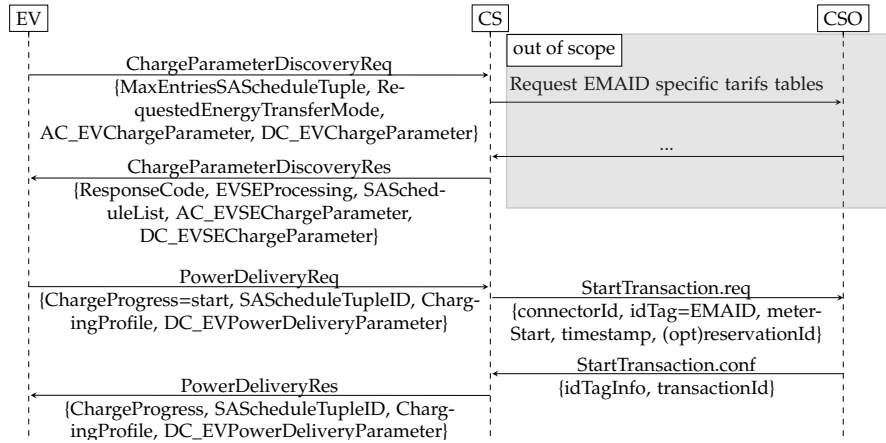


Figure 7.4.: Target Setting and Charge Scheduling

**Charge Control and Rescheduling** During the charging process, a constant status loop maintains the communication (Figure 7.5). In this sequence, the EVCC periodically transmits the *ChargingStatusReq*. Depending on the implementation, the CS may be triggered to send the current *meter value* to the CSO backend using the *MeterValues.req*. It should be kept in mind that sending the meter values to the CSO backend may enable external attackers to identify EVs by analyzing their charging behavior based on the consumed energy. In the event that the SECC deems it necessary to receive a metering receipt, the *ReceiptRequired* flag is added to the response message. This response also contains the current *meter information* and the *EVSEID*. The EVCC will request a metering receipt in the following. This request is the *MeteringReceiptReq*, including the *SessionID*, the *signed meter values*, and the current charging schedule it previously received and checked for plausibility. Upon receiving the receipt, the CS may report the meter values to the CSO backend using the *MeterValues.req*. As previously stated, depending on the frequency of the *MeterValues.req*, it may be misused to identify EVs and should thus be avoided.

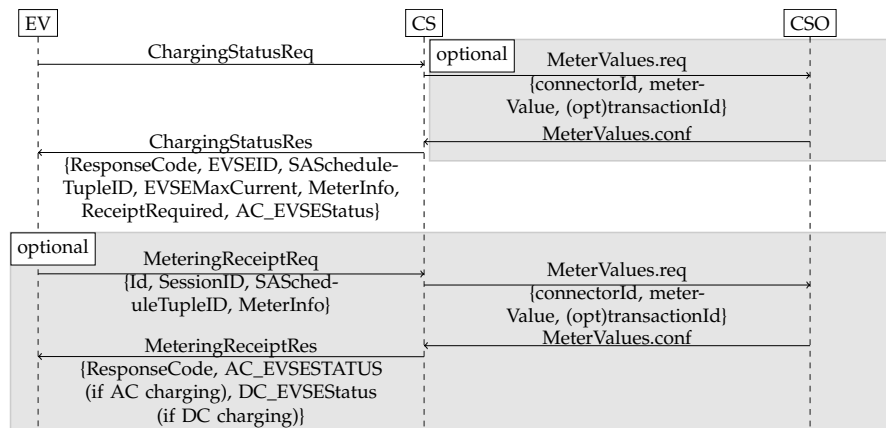


Figure 7.5.: Charge Control and Rescheduling

**End of the Charging Process** To end the charging process (illustrated in Figure 7.6) and instruct the SECC to terminate the voltage from the power outlet, the EVCC transmits a *PowerDeliveryReq* message with the appropriate *stop* flag in the *ChargeProgress* parameter. After this optional step, the EVCC sends the *SessionStopReq*, and the CS sends an *Authorize.req* containing the *EMAID* to the CSO backend to authorize the EV to stop the transaction. Once authorization is received by the CS, a *StopTransaction.req* request is sent to the CSO backend. This request contains the current *meter value*, a *timestamp*, the current *transactionID*, a *reason*, and optionally *EMAID* and *transaction data*, which in this case could be the *metering data*. Depending on the implementation, the CS can alternatively report the meter values using the *MeterValue.req* or *DataTransfer.req*. Once the CSO backend receives the request, it sends an *eRoamingAuthorizeStop* message to the CCH. This message contains the *SessionID* and the *EMAID*. This message is forwarded to the MO, which sends the *eRoamingAuthorizationStop* message via the CCH to the CSO backend. This message contains the *EMAID*. The CSO backend sends a *StopTransaction.conf* to the CS, which informs the EV whether the session has been stopped successfully. In parallel, the CSO backend sends the *eRoamingChargeDetailRecord* to the CCH, which forwards it to the MO. This message defines the CDR's contents, which must be sent for every session authorized over the CCH [142]. The CCH stores all reported CDRs and archives them. The CDR contains *SessionIDs*, *ProductIDs* for consumed products, the *EVSEID*, the *EMAID*, *timestamps* for start and end times of the charging session, start and end *values of the meter*, the *consumed energy*, a *signature over the metering data* and *Hubject identifiers* for the (CSO) and (MO) that participated in the roaming scenario.

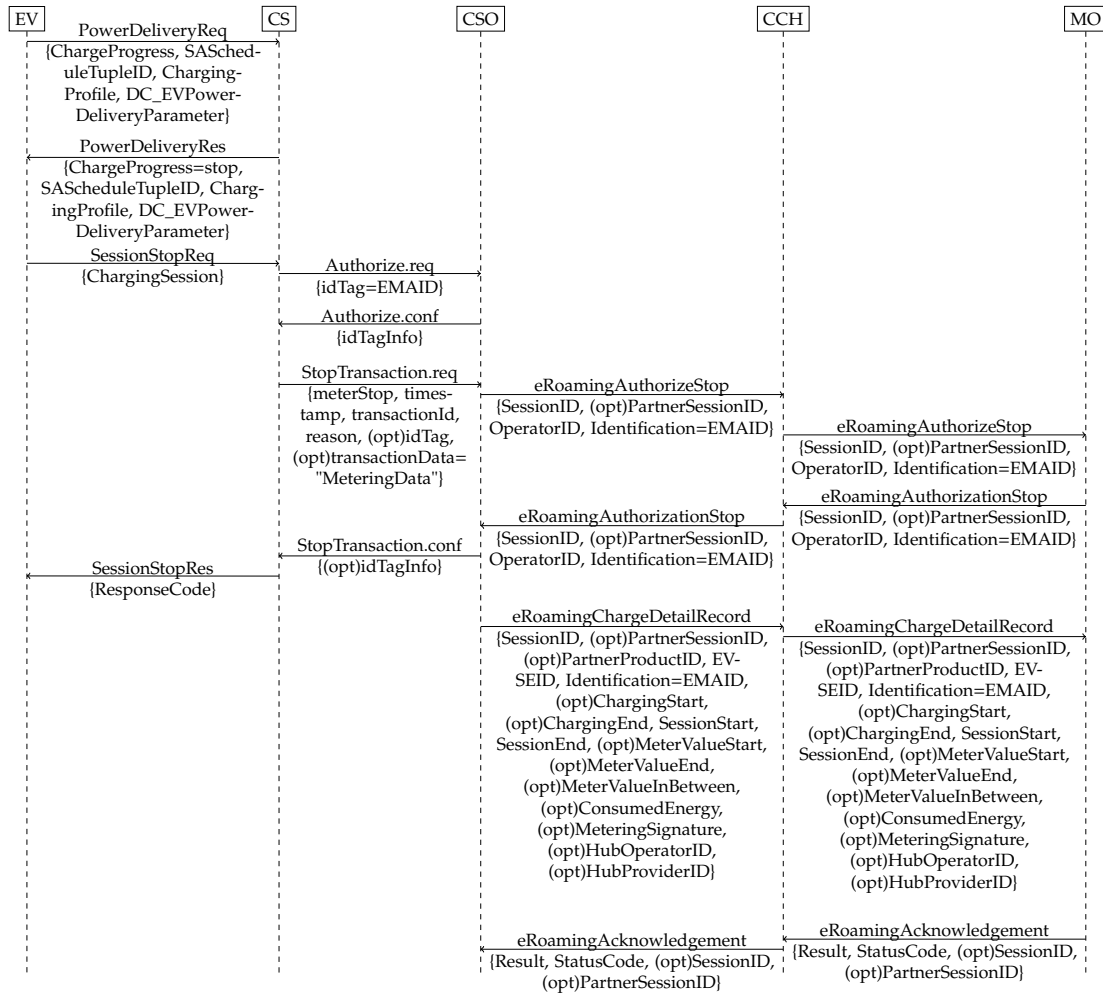


Figure 7.6.: End of the Charging Process

Table 7.1 summarizes the privacy-related information transported during the aforementioned scenario. First of all, multiple EV identifiers are involved. The EVCCID is a unique identifier for the EVCC of an EV throughout its lifetime. The EVCCID is transmitted at the beginning of every charging session to establish a network connection to a CS. It can be assumed that logging services on the CS may log the EVCCID, and thus, the CSO backend is highly likely to have access to this information as well. The OEM Provisioning Certificate contains the unique PCID and is used to identify an EV during the installation

of a Contract Certificate issued by an MO. It is installed during EV production and should remain constant for up to 40 years. We consider these out of scope for now as certificate updates and installation are not typically executed and are not well defined by the specification. The Contract Certificate containing the EMAID and used to identify the EV during charging sessions also remains constant for up to six months or until the contract with an MO is canceled. The EMAID may also be transported to all system stakeholders during the certificate update process. The EMAID additionally contains the identity of the MO and is thus transmitted to the CSO as well as to the CCH and the MO for billing purposes during the roaming scenario. Considering that an EV in the private sector is typically used by a minimal range of people, limited to the EV owner and his family members, these EV identifiers can be treated as personal identifiers as well.

Table 7.1.: Personal Data in PnC

	EV	CS/CSO	CCH	MO
CDR		✓	✓	✓
Charging Parameters	✓	✓		
Contract Certificate	✓	✓		✓
EMAID	✓	✓	✓	✓
EVCCID	✓	✓		
EVSEID	✓	✓	✓	✓
MeterID	✓	✓	✓	✓
Power Consumption	✓	✓	✓	✓
Time	✓	✓	✓	✓

Moreover, the identity of the CS is transmitted to all parties involved in the charging and billing process as well. This is done either via the EVSEID or the identifier of the meter in the CS. These values remain identical throughout the lifetime of a CS. For this reason, it is possible to create a map with the location of every EVSEID. Partially, this map already exists under <https://www.goingelectric.de/stromtankstellen/>, where people uploaded logs of charging processes containing EVSEIDs. Thus, the identity of a CS also reveals the location of a charging session and the respective EV using PnC. Likely, the EVSEID simply encodes the location of a CS in the prefix or postfix of the identifier, as the EVSEID is not specified to be random.

Parameters for the charging session, like the decision between AC and DC charging, are only transmitted to the CS but not forwarded to other parties. Power consumption determines how much energy an EV has consumed. Especially if the CS reports meter



---

values to the CSO backend in regular and cyclic time intervals, this information can be used to analyze the charging behavior of the EV. Generating profiles based on this information makes it possible to identify the user through his EV, too. Timestamps are similar to Power Consumption but indicate when and how long the user charged his EV.

The CDR, a comprehensive record of the charging session, is a significant privacy concern for PnC. It consolidates most of the sensitive data described above and is distributed in the backend without the user's knowledge. Especially since the CCH, according to [142], stores and archives this information. The CDR contains all information necessary for user identification. It provides information about a charging location, time stamps, EMAID, EVSEID, and additional information about the CSO whose CS the user visited and the MO the user has a contract with.

Our analysis underscores the need for a privacy-preserving charging protocol. The current PnC standards make it transparent to every stakeholder (CSO, CCH, MO) involved in charging and billing EVs, who charges, when, and where. We further discuss which information would be sufficient when introducing our privacy-preserving charging protocol in Section 7.8.

## **7.4. V2G Attack Strategies**

An attacker must gain control over a substantial number of charging processes to disrupt the stability of the power grid and, in the worst-case scenario, achieve a blackout [212]. In this section, we show different attack strategies an attacker may choose, starting with attacks against CSs or EVs, charging communication, V2G processes, and malicious backends.

### **7.4.1. Attacks on EVs and CSs**

Attacks capable of assuming control over vehicles or CSs can execute an attack against the grid through a charger-bot or car-bot attack or by masquerading as a flexibility resource.

---

## **Charger-bot Attack**

In this scenario, the adversary succeeds in installing malicious firmware onto a CS, often through means such as exploiting a USB port or introducing a tampered update image (as discussed in Section 7.2). Social engineering tactics may be employed to gain physical access to devices. This enables the adversary to establish persistent remote access, permitting eavesdropping on the CS's communication with the CSO and the vehicle. The attacker can manipulate messages to induce the desired behavior, such as suppressing load reduction commands from the backend, customizing the charging schedule to align with the planned attack schedule, forging tariff information, or arbitrarily initiating and terminating charging sessions (note that in ISO 15118, only the vehicle can pause/resume a charging session). Additionally, the adversary can use the physical access gained to start or halt the charging of the connected car, for instance, by pressing a button on the user interface commonly found in many private and public chargers. Given that Alternating Current (AC) chargers intended for private use tend to have limited security features, they are susceptible to exploitation, facilitating such attacks.

## **Car-bot Attack**

In a car-bot attack, the adversary acquires (remote) access to a vehicle's EVCC or E/E system. This access may be gained using one of the exploits outlined in Section 7.2, allowing the attacker to inject commands to initiate/terminate or pause/resume charging. In the case of paused charging, the resumed session employs the previously negotiated charging schedule and parameters, which may not align with the current grid conditions and can result in a sudden peak in consumption. If not, the adversary can inject a new charging schedule, pausing and activating the load at a chosen time to facilitate the planned attack. Many OEMs offer connected car apps that enable users to manage battery recharge. The adversary can exploit compromised apps or stolen user credentials to create such schedules, e.g., by altering the departure time. Furthermore, the adversary can connect their prepared EVs to the local grid using nearby public charging CSs to amplify the impact caused by the botnet.

## **Masquerading as a Flexibility Resource**

The adversary can connect a vehicle simulator like EVSim [40]. It presents itself as a controllable load to the charger but provides false data about power consumption while

---

disregarding any overload or control signals.

## **Manipulating PKI**

An adversary controlling parts of a PKI can use Online Certificate Status Protocol (OCSP) messages or Certificate Revocation List (CRL) to invalidate all certificates in the charging infrastructure at a certain point in time. This blocks all vehicles from further charging attempts. The attacker may also sign their own certificates, allowing further attacks against certain protocols. This last attack, however, is considered the most difficult for an attacker since certification processes are very secure in most cases, but [97] has shown several incidences happened at CAs in the past years.

### **7.4.2. Malicious Backend Attack**

Backend systems managed by CSO and DSO play a key role in load management by providing charging schedules that align with the local grid's prevailing conditions. Consequently, attacks targeting these backend systems have the potential to create critical grid conditions and must not be underestimated. If an adversary manages to gain control over the management processes within the backend, they can monitor grid load fluctuations and manipulate the charging behavior of managed chargers. The adversary can meticulously craft a charging schedule designed to induce overload and disseminate it to CSs via the OCPP. Each CS then initiates a renegotiation of the ongoing charging session through ISO 15118. In cases where the EV and charger had initially agreed upon dynamic control mode (as opposed to scheduled mode, where the EVCC adheres strictly to user preferences), the CS seamlessly adjusts its charging behavior following the backend's directives. Furthermore, the adversary can inject tariff updates, exploiting the fact that authenticating signatures is optional. These backend systems can encourage or discourage EVs from initiating or continuing charging sessions by manipulating energy prices.

It is worth noting that backend systems like these typically require ISO 27000 certification in Europe, imposing stringent security measures that render them more resilient to compromise compared to private charging facilities. However, our research primarily focuses on the latter.

---

### 7.4.3. Attacks on Charging Communications

Attacks that target the communication channels of the charging process can lead to various disruptive outcomes, for example, DoS attacks, session resumption attacks, MitM attacks, and injection and authorization attacks against PLC. In the following, we explain these possible attacks.

#### DoS Attack

To abruptly halt charging in a specific area, an adversary can deploy a jamming signal, for example, as demonstrated in the Brokenwire attack [105]. If the charging system automatically recovers, the attack can be repeated. Furthermore, DoS attacks on the connection to the backend via a cellular network by jamming the signal or setting up a fake base station, assaults that effectively disrupt load management processes in the targeted area.

#### PLC Injection Attack

Given that the link layer in ISO 15118 is not protected against eavesdropping and manipulation, an adversary can employ available commercial devices, such as CCS Listener [188], or custom-made PLC devices [48], connected to the same grid through methods like an earthed socket on the charger or at the parking lot. These devices can capture, inject, and modify V2G messages. Since TLS is optional in ISO 15118-2, all communication can be exposed to eavesdroppers. The adversary can gather information such as the EVCC identifier (vehicle's MAC), the *sessionID* of the current charging session, charging schedules, and tariffs. Using the captured *sessionID*, the adversary can craft and inject arbitrary messages adhering to the ISO 15118 format to manipulate the charging session to align with their chosen attack strategy. Additionally, the adversary can disrupt the UDP-based pairing via HomePlug Green PHY (HPGP) between the EVCC and the SECC, resulting in disconnection, charge termination, or even forcing the vehicle to connect to a decoy.

---

## MitM Attack

Communication between a CS and the CSO backend typically occurs without encryption, making it susceptible to interceptions. Moreover, the absence of authentication between these systems permits adversaries to impersonate any of the involved entities and spoof communications. Even if TLS is employed, OCPP 2.0.1 does not mandate charger-side certificate validation when interfacing with the backend. Consequently, if the adversary infiltrates the communication link, for instance, through the use of a fake base station, the backend system can be easily spoofed. This enables attackers to remotely transmit OCPP commands to initiate/terminate charging or launch complex attacks involving charging schedules and tariffs. Spoofing the charge point is also feasible in ISO 15118-2/20, as certificate validation is relaxed in private environments. In certain instances, obtaining the leaked private root may be necessary for success in ISO 15118-20.

## PnC Authorization Attack

By utilizing an in-cable adapter and exploiting vulnerabilities in channel binding and PnC authentication (cf. Section 7.3), the adversary can impersonate a legitimate EV to gain access to the grid. Given that charging schedules often rely on the reputation-based behavior of authorized consumers, the adversary can exploit priority charging options using this method.

## Session Resumption Attack

Once authenticated, a vehicle can pause and resume ISO 15118 sessions by presenting the *sessionID* identifier. In EIM mode, V2GTP communication lacks encryption, rendering all parameters accessible to potential attackers. Other means of obtaining this value include compromising the EV or charger or guessing the value in a brute-force attempt. Some chargers employ simple counter values for *sessionID*, which makes the session ID more predictable. The length of this identifier also depends on the implementation; in the open-source implementation RISE-V2G, this ID has only 8 bytes. Using the *sessionID*, the adversary can take control of the session, inject messages, or impersonate the EV.

---

## 7.5. Attack Examples against Charging Protocols

In this section, we present two example attacks, providing detailed technical insights to demonstrate how attackers can manipulate the grid by targeting charging communication using the previously outlined attack strategies. First, we discuss a MitM attack on OCPP, followed by an explanation of a combined attack involving PLC and ISO 15118 communication.

### 7.5.1. OCPP Charging Schedule Attack

An attacker can potentially compromise OCPP communication between a charge point and its operator's backend system, even assuming a MitM role. This attack requires solving several challenges to gain access to the network and the communication channels.

The attacker must initially gain access to the network utilized for charging communication. CSs are typically connected either physically to a home or company network or remotely through a cellular network. In our tests (cf. Section 7.6), we maintained access to the company network, which served eight AC chargers and one DC charger, via a physical LAN connection. Alternatively, a WiFi or cellular connection could be employed.

A cellular connection by itself would not protect against an attacker due to the well-known security vulnerabilities in existing mobile communication standards like GSM [1], SS7 [107], UMTS [126], and LTE [157, 78]. The absence of authentication and integrity protection enables attackers to locally eavesdrop on or redirect mobile traffic from CSs through a false base station [131, 17, 166]. Although using a VPN could mitigate this attack, we did not observe this technique in practice during our analysis.

An attacker can employ ARP spoofing to redirect traffic to the attack's device. At the application level, the attacker needs to bypass the TLS protection. It is important to note that TLS is often optional or inactive in private environments and more affordable wallboxes. Furthermore, certificate validation is often skipped even if TLS is active, as we show in Section 7.6.1. Consequently, presenting a self-signed certificate during the key exchange suffice to establish the MitM position.

As a subsequent step, the attacker must send a thoroughly crafted *SetChargingProfileRequest* message with a new charging profile to the charge point, reducing the charged power to a minimal level (cf. Figure 7.7). An attacker can also reset the limit within this profile at an arbitrary point in time. In this manner, the adversary can manipulate the charging

---

behavior of the connected EV, ensuring that its battery is not fully charged until the planned start of the attack targeting the grid (for example, to cause a blackout).

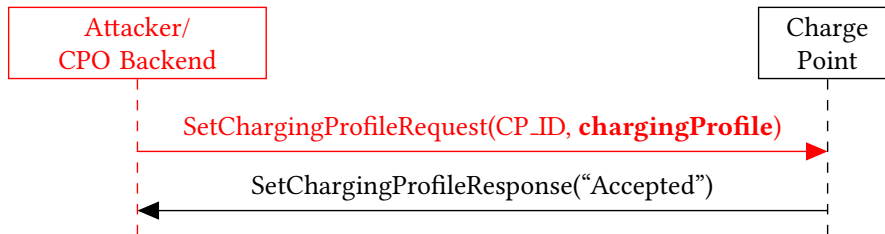


Figure 7.7.: Manipulating OCPP communication

Once network access is established, this attack is straightforward to execute. The described attack steps proved effective for all evaluated charge points in Section 7.6.1.

### 7.5.2. ISO 15118 MitM Attack

Given that the low-level PLC in ISO 15118 is unencrypted [91], we exploit it as an entry point to compromise load management during EV charging in the setup described in Section 7.6.2. The attack on HomePlug Green PHY (HPGP)/PLC unfolds in three steps: (1) connecting to the PLC network, (2) intercepting messages to position as MitM, and (3) executing an application-level attack.

As the first step, the adversary connects to the HPGP/PLC network of the targeted charger to impersonate the charger for a vehicle. In ISO 15118, the Signal Level Attenuation Characterization (SLAC) protocol establishes pairing between an EV and a CS [91]. In this process, the CS serves as the central coordinator that broadcasts *CM\_SLAC\_MATCH* messages containing the *Network Membership Key*. Any device seeking to connect to this network requires this key.

Once the adversary gains access to this PLC network on the MAC and IP levels, an attack against the service discovery (SDP)[90] of ISO 15118 becomes possible. Figure 7.8 illustrates this process, where an EV initiates a search for the connected CS using an *SECC\_RequestMessage*. Both the CS and the MitM attacker respond with an *SECC\_ResponseMessage* containing the target port and IP address.

To expedite the message transmission faster than the charger, we implemented a very efficient native program in C. This program can also disable the TLS connection for the EV

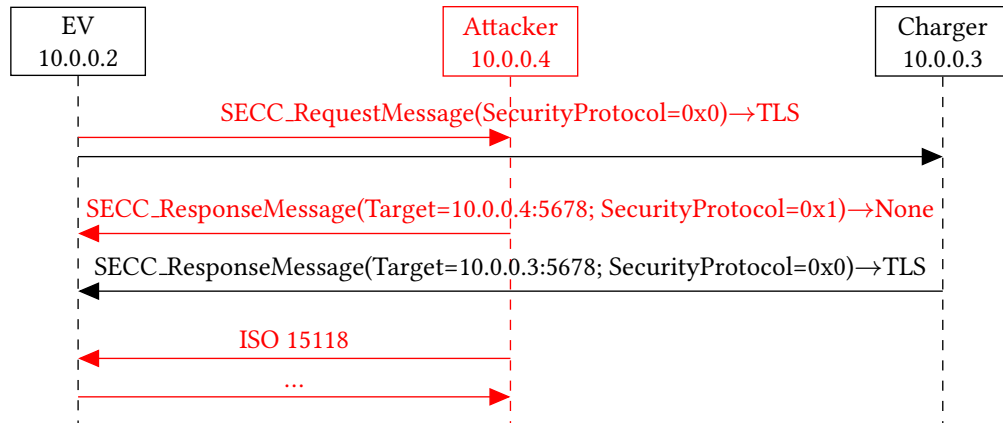


Figure 7.8.: MitM attack on ISO 15118 communication

by injecting an SDP message that convinces the vehicle that the charger does not support TLS (and simultaneously convinces the charger that the vehicle did not request TLS). This downgrades authentication options to EIM only, facilitating plaintext communication. Since ISO 15118-2 management operations are also possible with EIM, this should suffice for most adversarial goals.

With network access established, the attacker can redirect traffic from the vehicle to their own device by sending a specially crafted response with their port and address. In this position, the attacker gains control of the charging session with the EV, allowing changes to be made to various parameters of the charging process without further difficulties. For instance, the attacker can inject a *SessionStop* request to terminate the charging session without triggering an error state or inject a *PowerDelivery* request to initiate the renegotiation of charging parameters and enforce the desired charging schedule. This enables the attacker to ensure that the EV remains available as a load when needed.

Similarly, the attacker can impersonate the vehicle to the charger. Notably, all application messages in ISO 15118's communication flow are encoded using the Efficient XML Interchange (EXI) standard. EXI is a relatively uncommon standard, and only a few implementations exist. Due to the protocol's strict timeouts, we implemented a highly efficient EXI decoder and encoder in Rust that is more than ten times faster than the Java open-source implementation and thus allows us to respond fast in a MitM position.



---

## 7.6. Practical Security Analysis of V2G Communication

Based on the attacks described in the previous section (Section 7.5), we analyze the security with different CSs. In this section, we show these attacks influencing the charging process are possible. In particular, we evaluated the main communication links: CS↔CSO and CS↔EV. In the evaluation for communication CS↔CSO, our evaluation environment was a company network with three types of CSs from different manufacturers: two 22 kW AC wallboxes and one Direct Current (DC) high power charger (HPC). We are using the factory default security settings for every CS. Each implemented OCPP 1.6 and TLS to interface with the CSO (cf. Table 7.2). In the second practical evaluation, we analyzed the security of data exchange between one CS and the EV via ISO 15118.

### 7.6.1. Evaluation of OCPP Communication

We intercepted messages between the CS and the CSO in the company network to analyze the security of this connection. Therefore, we checked the TLS version, the supported ciphers, the certificate validation, and the possibility of injecting OCPP messages. All CSs accepted arbitrary self-signed certificates and thus, even with activated TLS encryption, were vulnerable to MitM attacks. With help from the manufacturer, we could configure the firmware to validate certificates correctly for Wallbox B. All CSs featured insecure TLS versions. The usage of SSL and ciphers containing RC4, in particular, pose a severe threat to downgrade attacks.

Table 7.2.: CS Configurations Observed in Experiments

Wallbox A	OCPP 1.5, 1.6	TLS 1.0, 1.1, 1.2	TLS active
Wallbox B	OCPP 1.6, 2.0	TLS 1.0, 1.1, 1.2; SSL 3.0	TLS active
HPC	OCPP 1.5, 1.6	TLS 1.0, 1.1, 1.2; SSL 3.0	TLS active

Using SteVe [158] for OCPP and our own self-signed certificates, we could read and modify the communication of all three products to cause the desired behavior. This enables the attack to the charging schedule as described in Section 7.5.1. We can interrupt the charging process at will by altering the charging schedule with a carefully crafted message injected into the communication. Figure 7.9 shows Wallbox A disabled by a remote attack on the OCPP connection, indicated by the red status light.



Figure 7.9.: Wallbox disabled by OCPP Attack

### 7.6.2. Evaluation of ISO 15118 Communication

In the second practical evaluation, we analyzed the security of data exchange between CS and the EV via ISO 15118. To intercept and manipulate the low-level communication using PLC, we designed and built an evaluation setup based on a PLC development board with a Qualcomm QCA7000 controller shown in Figure 7.10. Our experiments used RISE V2G [133] as the reference implementation for ISO 15118-2. Figure 7.11 displays our evaluation environment where our test device is connected to a wallbox. We are connected directly to the charging station via USB to analyze the logs of the CS.

In a simple DoS attack, we were able to interrupt the active communication session and stop the charging process by using V2GInjector [143, 48] to send a `SessionStop` message from a rogue PLC device. Due to the strict timeouts and message format requirements in ISO 15118, delaying or injecting unexpected or malformed messages also led to charging failures.

Since the performance of Python-based V2GInjector was insufficient for more complex attacks, we developed a native C implementation for further tests as well as a high-performance EXI parser written in Rust. This way, we realized a MitM attack on ISO 15118 communication described in Section 7.5.2. With this, we could eavesdrop on charging sessions and collect data like private payment details or service information. These data help predict charging behavior and plan offensive actions. We could also inject our own charging schedules and tariffs to activate loads at a selected time as well as pause and restart charging sessions at will.

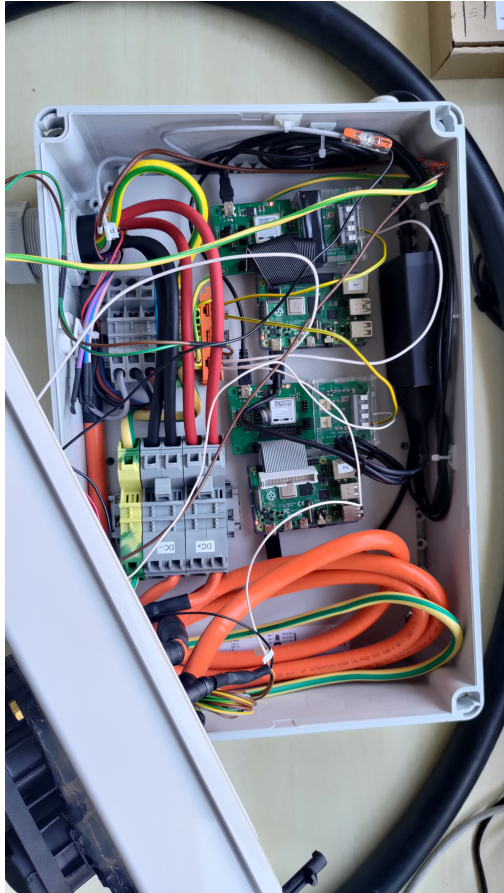


Figure 7.10.: Test Setup for ISO 15118 communication



Figure 7.11.: Evaluation Environment for ISO 15118 communication

---

### 7.6.3. Relevance of the Evaluation Results

Both practical evaluations show that it takes little effort for an adversary to control charging sessions. The relevance of this result for the security of power supply in the local grid partly depends on the real-world adoption of the protocols at hand. Therefore, we analyzed products from the German market overview [129], including wallboxes with 11 kW and 22 kW, AC and DC chargers. As shown in Figure 7.12, 73 out of 93 models support OCPP, but only two offer the newest version (according to the product information), protecting against MitM attacks. The vast majority supports OCPP 1.6 or lower with known specification weaknesses. Also, 31 CSs support a Modbus connection to connect locally to an EMS. ISO 15118 is implemented only in 6 CSs of the 93 products we compared. Considering this analysis and our practical evaluation, we think it is fair to say that most charging products should improve their measures against V2G attacks.

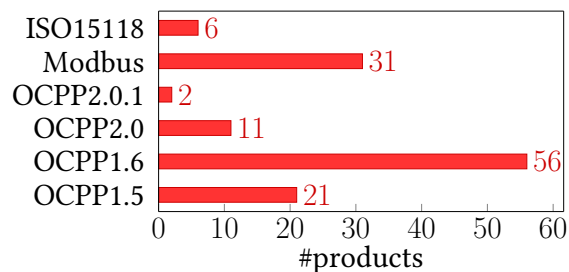


Figure 7.12.: Protocols supported in modern chargers

## 7.7. Limitations and Mitigation

This section covers the practicality of the presented attacks to influence a local power grid. Additionally, viable mitigation strategies to effectively counteract the identified vulnerabilities are discussed.

### 7.7.1. Practicality of the Attack

Based on our simulations in [212], it is feasible to damage equipment or trigger a blackout in a local grid, but this requires accumulating a substantial load over time. If these load

---

---

limits are exceeded, the intervention of the DSO becomes necessary to prevent damage to equipment, blackouts, and potential harm to consumers.

Our attack to impact grid stability assumes a detailed knowledge of local EV models and installed chargers. However, the currently fragmented market may consolidate in the future. Adequate detection by DSO or CSO could potentially prevent an attacker from experimenting over an extended period to find the correct timing. Nevertheless, practical experience with fire drills or power outages suggests that this approach is unlikely to work unless regularly exercised.

Larger-scale attacks and generalizations must also be considered. Future attackers might create a "blackout botnet" comprising several components to reach the required load, denoted as  $L_{total} = L_{basic} + L_{delayed} + L_{scheduled} + L_{controlled} + L_{blackiot}$ :

- $L_{basic}$  represents normal energy consumption in the neighborhood driven by the residents' personal needs.
- $L_{delayed}$  includes consumers whose charging was shifted due to a DoS attack like Brokenwire [105], which can interrupt multiple charging sessions within a radius of up to 40 meters.
- $L_{scheduled}$  accounts for additional load by a predictable or attacker-manipulated charging schedule. This can be achieved through various means described in Section 7.4.3, 7.6, and 7.5.
- $L_{controlled}$  includes consumers either belonging to the adversary or under their direct control, for instance, through compromised software updates [128, 32].
- $L_{blackiot}$  reflects the possibility of boosting the attack using a botnet of regular (non-managed) but high-wattage devices, such as those demonstrated in [170]. Inconspicuous devices like wallboxes, air conditioners, heat pumps, and other systems may remain compromised for an extended period since they are rarely updated or replaced.

Moreover, an attacker that has access to local networks of households could exploit local power generation capabilities, such as inverters for photovoltaic systems, by shutting them down to further increase stress on the grid.

---

## 7.7.2. Mitigation Strategies

Our simulations demonstrate that an adversary can potentially cause a local grid to collapse by manipulating V2G communication used for load management. To counteract these threats, grid operators, policymakers, and industry stakeholders should consider the detailed conditions for a successful attack, the required effort, and the possible impact. Several mitigation strategies can be employed:

- *Fix Weaknesses*: Identified vulnerabilities in deployed protocols and implementations should be promptly addressed. Strong mutual authentication and the mandatory use of secure channels on all communication layers between EVs and chargers, as well as chargers and backends, would help mitigate the identified MitM attacks.
- *End-to-End Authentication*: Management data and commands transferred via intermediaries should be authenticated end-to-end. This would prevent adversaries from manipulating charging schedules or tariffs.
- *Network Security*: Protecting the underlying communication networks and separating network segments used for load management from other types of traffic is crucial, especially in private environments where insecure protocol versions may persist in legacy systems.
- *Countermeasures for Known Attacks*: Countermeasures should be implemented against widely applicable attacks, such as Brokenwire [105], to prevent attackers from combining multiple vectors.
- *Security Processes*: Participants in the critical infrastructure should establish security processes to ensure a minimum level of configuration and security management. While automotive ISO/SAE 21434 mandates this for car manufacturers, it should be extended to other participants in the V2G infrastructure.

By implementing these mitigation strategies, the V2G ecosystem can significantly reduce the risk of successful attacks and enhance the grid's overall security. However, these techniques do not fix the privacy issues shown in Section 7.3.3. For this reason, we develop a privacy extension in the following section, starting with a requirements analysis.

---

## 7.8. Privacy-Enhancing Charging Solution

Despite the security fixes and recommendations discussed earlier, the privacy concerns raised in Section 7.3.3 persist. It is important to note that privacy violations extend beyond malicious actors, encompassing non-malicious insiders, such as PnC service providers like acCSO, CCH, or MO, who may have legitimate access to PnC users' personal data. This includes the common practice of misusing user data for various purposes, including selling them to third parties such as insurance companies or advertising agencies [35]. After we outline the requirements and propose a protocol extension, we discuss the security and privacy properties of our proposed solution.

Despite the security fixes and recommendations discussed earlier, the privacy concerns raised in Section 7.3.3 persist. It is important to note that privacy violations extend beyond malicious actors, encompassing non-malicious insiders, such as PnC service providers like CSO, CCH, or MO, who may have legitimate access to PnC users' personal data. This includes the common practice of misusing user data for various purposes, including selling them to third parties such as insurance companies or advertising agencies [35].

After we outline the requirements, and propose a protocol extension, we discuss the security and privacy properties of our proposed solution.

### 7.8.1. Requirements Analysis for a Privacy-Enhancing Charging Solution

In this section, we define requirements and assumptions for a privacy-preserving PnC solution based on a privacy analysis of the system architecture introduced in Section 2.3.1.

#### Minimal Data Model

To establish privacy-focused requirements, we must assess the system's stakeholders, roles, and components and determine the essential data required for each role to fulfill its functions. The data minimization principle must be upheld, ensuring that each role accesses only the necessary data to safeguard customer privacy. The entities in the charging process and their part in the data processing are:

- *EV User*: The user should be able to verify charging session details, including location, time, and energy consumption.

- 
- *CS*: To prevent MitM attacks, the CS needs to establish a secure and confidential channel with the EV and maintain it throughout the charging session. It must also collect data on energy consumption, charging session timing, and billing-related information.
  - *CSO*: The CSO needs to know which MO should receive billing data, among other potentially relevant information, based on specific usage conditions.
  - *MO*: The MO requires information about the energy consumed and which customer should be billed.
  - *CCH*: The CCH only needs information regarding the designated MO for communication without any further customer or session data.

### Requirements for Privacy-Preserving Charging

When designing our privacy-enhancing solution, we establish privacy protection objectives based on the European GDPR [53] and our attacker model outlined in [sec:attackModel]. We consider seven standard privacy protection goals from [70, 182], which we map to the PnC architecture as follows:

- P<sub>1</sub> Unlinkability*: Charging sessions conducted by a customer must be unlinkable. Achieving this involves minimizing data, adhering to the principle of data purpose, and providing anonymity or pseudo-anonymity.
- P<sub>2</sub> Accountability (of Anonymous Users/Pseudonyms)*: The MO must be able to bill the correct customer for their charging session.
- P<sub>3</sub> Non-Repudiation*: Users should not be able to dispute charging sessions to evade payment.
- P<sub>4</sub> Authorization*: Access to the charging service should only be granted to registered customers of the relevant CSO or MOs participating in roaming agreements with the CSO.
- P<sub>5</sub> Integrity*: Billing information provided to the MO must be accurate and tamper-proof.
- P<sub>6</sub> Confidentiality*: Data such as tariffs, billing details, and technical information (e.g., keys) must remain confidential and protected from unauthorized access.
- P<sub>7</sub> Data Minimization*: Each entity should only have access to the required information, which is given in our minimal data model in the previous section.



---

While specifications of GDPR regulations may vary by country, our focus here is on these specified goals, with legal compliance being beyond the scope of this work.

Regarding confidentiality, it is essential to note that if attackers can extract the Contract Certificate or OEM Provisioning Certificate along with their private keys, they could impersonate customers, charge other EVs on their behalf, and even request new Contract Certificates. Leakage of keys and certificates can occur through side-channel attacks or malicious firmware on the EV. Ideally, private keys should be generated securely within the vehicle, never leaving it. Cryptographic keys should remain secure, even during use, by establishing a secure execution environment for cryptographic operations, ideally with hardware separation to prevent software-based leakage. Revocation mechanisms should be in place for compromised keys. The proposed solution should also minimize communication and computational overhead while ensuring data security.

The following list summarizes these requirements:

- R<sub>1</sub> Secure Key Generation:* Private keys should be generated on the vehicle to prevent their disclosure during their generation and installation process.
- R<sub>2</sub> Secure Key Storage:* Private keys should be stored in a protected and secure environment to prevent their leakage, even through side channels.
- R<sub>3</sub> Secure Execution Environment:* To ensure cryptographic keys are not leaked when they are used, a secure execution environment should be provided, ideally physically separated from the hardware components.
- R<sub>4</sub> Revocation:* The backend needs to be able to revoke compromised keys and exclude EVs to prevent misuse.
- R<sub>5</sub> Protected Firmware:* Firmware needs to be protected from modification to prevent information leakage.
- R<sub>6</sub> Minimal Communication Overhead:* Keep the additional required protocol and message overhead as small as possible.
- R<sub>7</sub> Minimal Computational Overhead:* Keep the computational overhead as small as possible.

---

## Assumptions

We make several assumptions regarding our privacy-preserving PnC solution and discuss their implications:

- *Data Accuracy:* We assume that the data received by EVs from CSs are correct. While CSOs can manipulate their CSs' data, regulations, and calibration authorities (in Germany) ensure data accuracy. Alternatively, users can use in-cable metering systems<sup>1</sup> for control.
- *Data Exchange Between CSO and MO:* We assume that CSOs and MOs may exchange raw data related to charging sessions using channels other than our privacy-preserving solution. In this case, an MO cannot be prevented from location-based user profiling. While this threat persists, customers and MOs have agreements obliging MOs to protect customer privacy. Therefore, the primary focus should be protecting personal data from CCHs and CSOs, who lack direct agreements with PnC users.
- *TPM 2.0:* We assume that EVs are equipped with a TPM 2.0. This TPM is used to securely generate and store keys, provide a secure execution environment, support key binding to the system state, and attest to the system state in a privacy-preserving manner.
- *CS Online Availability:* We assume that CSs are always online, although ISO 15118 supports both online and semi-online CSs [90].

Considering these assumptions, we present a robust privacy-enhancing charging solution in the next section that addresses the identified privacy concerns while regarding the established standards.

### 7.8.2. Proposed Privacy Extension for EV Charging

In Section 7.3.3, our data flow analysis revealed that sensitive personal information is transmitted to all entities involved in the PnC process chain during the EV charging process.

Our proposed solution involves incorporating a TPM within the EVCC or another central component of the EV. This TPM's primary objective is to guarantee this critical component's

---

<sup>1</sup><https://www.ubitricity.com/en/>

---

integrity. Furthermore, the TPM enables the implementation of DAA, which facilitates anonymous authentication when interacting with third parties. In this context, the EV takes the role of the Host, equipped with a TPM, each CS functions as a Verifier, and the MO acts as an Issuer responsible for creating a group for its customers.

We anticipate a growing adoption of TPMs in vehicles, especially since manufacturers such as Infineon have released automotive-qualified TPMs [83]. Additionally, the utility of TPMs has already been discussed in related scenarios, such as using a TPM to secure over-the-air updates for automotive head units [60].

The remainder of this section outlines our proposed solution. First, we detail the initialization steps necessary to augment the charging and billing protocols with DAA. Subsequently, we describe the protocol modifications required for ISO 15118, OCPP, and OICP.

## Initialization

The conventional ISO 15118 protocol mandates a setup phase where each MO possesses a certificate authorized through conventional trust chains by Certificate Authorities (CAs). The MO uses the private key of this certificate to sign contract certificates for each EV. Given our introduction of DAA for EV authentication, modifications to these certificates become necessary. While the general CA structure remains unchanged, the MO certificate now incorporates their DAA public key ( $DAApk_{MO}$ ), generated during the DAA setup phase outlined in Section 7.1.

Before initiating a charging process, EV owners and MOs must conclude a contract. The DAA join protocol is utilized to establish a contract between EVs and MOs. During this phase, the TPM in the EV generates a DAA private key  $DAA sk_{TPM}$  for authenticating as a group member of the MO. An EV can perform this process with its MO locally or via a secure channel over an online CS.

## Extension of ISO 15118

Our proposed protocol design emphasizes minimal data exposure to participants in the charging process while simultaneously safeguarding against dishonest parties. An overview of the protocol is presented in Figure 7.13.

ISO 15118 communication can commence as specified in [90]. The initial significant communication step is establishing a TLS channel between the EV and CS. During the

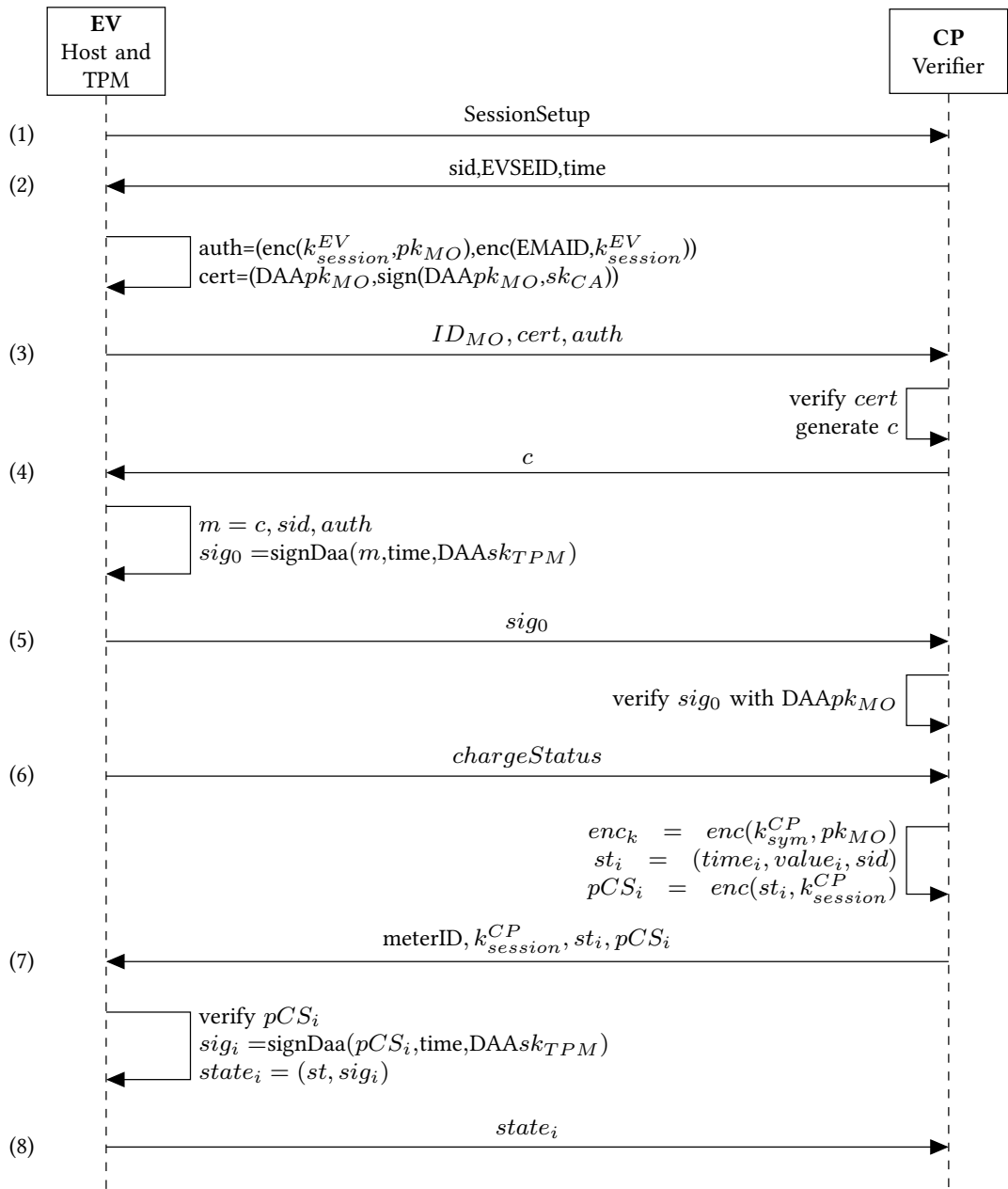


Figure 7.13.: Anonymous Charging Protocol

---

TLS handshake, the CS presents its certificate and certificate chain, verifying possession of the respective private key. By validating the certificate chain, the EV can confirm its communication with an authentic CS.

The *SessionSetup* step (1) of the ISO 15118 protocol, which is essential for initiating a charging session, requires modification. In the regular protocol, the EV requests to establish a session with the EVCCID, uniquely identifying the vehicle. However, this identifier is typically unnecessary, and we recommend omitting its transmission. The response of the CS (2) containing the session ID (*sid*), EVSEID, and timestamp (*time*) can remain unaltered.

The *ServiceDiscovery* and *ServiceDetail* message pairs, related to optional services, are not affected by the protocol changes we propose. Similarly, the *PaymentServiceSelection* message pair remains unchanged since it only encompasses payment options selected by the EV. Optionally, the *CertificateInstallation* and *CertificateUpdate* steps may be used to request new Contract Certificates. However, these optional steps are not considered and can be skipped during a standard charging session.

Next, the EV sends the *PaymentDetailsReq* (3), which typically includes the EMAID and Contract Certificate. In ISO 15118, this message contains personal data that needs anonymization in our protocols. The EMAID is truncated to the first five characters, identifying the MO as  $ID_{MO}$ . For future use in authentication, the EV transmits the public DAA key  $DAApk_{MO}$ , signed by a CA. Furthermore, the EMAID is encrypted using a fresh symmetric key ( $k_{session}^{EV}$ ) for charging session accounting. This symmetric key is then encrypted with the public key of the MO ( $pk_{MO}$ ). The CS validates the certificate *cert* and generates a challenge nonce (*c*) for the EV, which is conveyed in the *PaymentDetailsRes* message (4).

Subsequently, the EV sends the *AuthorizationReq* (5), which contains the signed challenge  $sig_0$  to authorize the respective private key of the certificate. Our protocol replaces the signing algorithm with DAA. The signed message incorporates the challenge *c*, session ID *sid*, and encrypted EMAID. The timestamp *time* serves as the basename, ensuring linkability throughout the charging session. The DAA secret key is unsealed only if the TPM confirms the integrity of the EV. After verifying the signature *sig*, the CS authenticates the EV and grants authorization for energy charging, confirmed through the *AuthorizationRes* response message.

The CS and EV then negotiate charging parameters and start the power delivery process. The EV periodically requests charging status (6), answered with *ChargingStatusRes* (7), including the MeterID, which uniquely identifies the meter within the CS, the current

---

timestamp  $time_i$ , and the current meter reading  $value_i$ . These values are now transmitted encrypted with a symmetric key ( $k_{session}^{CP}$ ). This key is further encrypted using the public key of the MO ( $pk_{MO}$ ). The EV verifies the encrypted message  $pCS_i$ . This encryption guarantees exclusive read permission for the MO. The ID of the meter  $meterID$  is also transmitted, as this value is not forwarded with the encrypted data to the CCH and MO. If necessary, the EV can sign ( $pCS_i$ ) using DAA (8), contingent on a *MeteringReceiptReq* requested by the CS. The CS can initiate this request by configuring the *RequestReceipt* field of the *ChargingStatusRes* message. The basename remains the initial timestamp ( $time$ ), ensuring linkability for all *MeteringReceipts* generated during the same charging session. The *ChargingStatus* and *MeteringReceipt* message pairs continue until the *SessionStop* message pair concludes the charging session.

### Extension of OCPP and OICP

The backend protocols OCPP and OICP serve two primary functions: They transmit the authentication tag  $auth$  to the MO via a CCH, already integrated into the protocols. The MO can then decrypt  $auth$  using its private key and respond, confirming using a valid EMAID to request charging authorization. The second part that is changed is transmitting the charging status to the MO via a CCH. These values are now transmitted encrypted ( $enc_k$  and  $pCS_i$ ) instead of transmitting raw values. This modification allows the transmission of energy consumption data to the MO while preventing the CCH from accessing these values. Notably, the *MeterID* allows locating the CS and thus also allows tracking the EV during the charging process. Additionally, the transmission of CDR is no longer necessary.

### 7.8.3. Security Discussion on the Anonymous Charging Protocol

In this section, we delve into the security and privacy implications of the protocol enhancements outlined in Section 7.8.2 focusing on how they align with the security requirements outlined in Section 7.8.1. We also address the broader security aspects of our approach.

**Achieving Unlinkability** One of the key security requirements ( $P_1$ ) is achieving unlinkability of charging sessions. We attain this by implementing DAA and employing distinct basenames for each new charging session, which allows users to stay anonymous for CS, CSO, and CCH without relying on pseudonyms ( $P_1$ ). [193] proves the unlinkability of DAA using a tamarin model. Consequently, we reduce the data exchanged during the

---

---

charging process to a minimum ( $R_6$  &  $P_1$ ), and only purpose-bound data are collected ( $P_1$ ). Classic identifiers like EVCCID and conventional contract certificates ( $P_1$  &  $R_6$ ) are no longer transmitted. The EMAID is encrypted, ensuring only the MO can access it ( $P_1$  &  $P_6$ ). While we maintain the identity of the MO and specific charging parameters accessible to some parties, the anonymization process minimizes the risk of personal identification ( $P_1$ ). It is worth noting that full unlinkability from the MO is unattainable due to the MO's need for customer accountability ( $P_2$ ). To prevent linkability via the mac address of a vehicle, we further suggest randomizing this address.

**Ensuring Non-Repudiation and Integrity** Accountability is ensured by including the EMAID of the Contract Certificate in the metering receipt, encrypted with the MO's public key ( $P_2$  &  $P_6$ ). This design choice preserves customer anonymity ( $P_1$ ) while supplying the MO with the essential billing data ( $P_2$ ). The requirement for non-repudiation ( $P_2$ ) is met through the signing of metering receipts using a secret known only to the EV. This approach also guarantees data integrity ( $P_3$  &  $P_5$ ).

**Ensuring Authorization** Authorization ( $P_4$ ) is effectively established by following the join protocol presented in Section 7.1. This mechanism prevents unauthorized or untrustworthy customers ( $R_4$ ) from gaining access to the Issuer's (MO) group.

**Ensuring Confidentiality** Our protocol design upholds confidentiality ( $P_6$ ) by encrypting all transmitted data. This encryption provides robust protection against unauthorized access to sensitive information. Moreover, the introduction of a TPM enhances key management ( $R_1$  &  $R_2$ ), secure execution of cryptographic operations ( $R_3$ ), and binding the use of keys to the state of the system ( $R_4$  &  $R_5$ ).

**Addressing Potential Threats** We also consider potential threats to our protocol. Owners of EVs may attempt to manipulate the protocol to either reduce metered energy consumption or change the EMAID for unauthorized charging. The reduction of charged energy is rendered impossible since this metric is verified solely by the EV and calculated by the CS. Changing the EMAID requires manipulation of the EV, a process prevented by the TPM's integrity checks ( $R_5$ ). The TPM also serves as a safeguard against extracting DAA keys ( $R_1$ ,  $R_2$ ,  $R_3$ , &  $R_5$ ).

Table 7.3.: Personal Data in the Adapted Charging Process

	EV	CS/CSO	CCH	MO
CDR				
Charging Parameters	✓	✓		
Contract Certificate	✓	✓		✓
EMAID	✓			✓
EVCCID	✓			
EVSEID	✓	✓		
MeterID	✓	✓		
Power Consumption	✓	✓		✓
Time	✓	✓		✓

**Minimizing Data Flow** Minimizing data flow is a critical requirement of our protocol changes ( $R_6$ ). These changes have substantially reduced the volume of data exchanged during the charging process. Table 7.3 provides an overview of the extent to which various types of personal data are accessible to different entities within the system. This matches the minimal data model defined previously ( $P_7$ ). Our modifications have effectively limited the accessibility of identifying information of EVs to MOs ( $P_1$ ,  $P_2$ , &  $P_3$ ) and minimized the data shared with the CCH. Also, the ID of the CS and the parameters for the charging process are limited to the CSO. Power consumption and time are values that can be accessed by CSO and MO but no longer by the CCH ( $P_1$ ). Both CSO and MO need this data for billing purposes ( $P_2$ ). The location of the charging session as well as the CDR are also not necessary for the charging process and thus are not transmitted ( $P_1$ ). The MeterID and EVSEID, which can identify the location of a charging process, are only shared between EV and CSO but no longer forwarded to CCH or MO. The MO receives the ID of the CSO instead ( $P_1$ ).

**Minimizing Impact on Protocols and Architecture** Our objective was also to minimize the impact of our protocol changes on existing standards and architectural frameworks. We achieved this by ensuring that the message sequences of ISO 15118, OCPP, and OICP remain unaltered. The primary changes in ISO 15118 affect four message pairs: *PaymentDetails*, *Authorization*, *ChargingStatus*, and *MeteringReceipt*. The *PaymentDetails* now contains a certificate with a DAA-based public key and an encryption of the EMAID. The *Authorization* signs the challenge using DAA. Similarly, the *MeteringReceipt* is signed. The response for *ChargingStatus* needs to perform an encryption of the consumed energy to create a receipt. Overall, ISO 15118 needs two additional encryptions for EMAID



---

---

and receipt and introduces DAA to perform signing operations at the same place where ISO 15118 previously performed RSA/ECC signatures. Importantly, both OCPP and OICP can transmit the encrypted receipt and EMAID without necessitating further adjustments to their protocols ( $R_6$  &  $R_7$ ). Although introducing TPMs in EVs may incur additional costs, it also opens up opportunities for enhancing vehicle security beyond the charging use case.

## 7.9. Comparison with State of the Art Anonymous Charging

In their work [77], Höfer et al. introduced the Popcorn protocol, which focuses on privacy-preserving charging based on ISO 15118. Popcorn utilizes an Identity Mixer for anonymous authentication of EVs. However, the protocol relies on an older version of ISO 15118 that lacks CCHs. Additionally, it introduces two additional actors, one for energy provider payments to MOs and another to address miscellaneous issues. Popcorn's authentication mechanism carries a significant overhead, requiring over a minute of computation time on a Raspberry Pi. Subsequently, Li et al. improved the computation time to a mere 20 milliseconds in their work [114] by introducing a new authentication scheme. Fazouane et al. suggest in [55] a protocol enhancement for Popcorn to provide provable unlinkability between distinct charging sessions.

Another protocol, proposed in [210], enables the reservation of CSs and dynamic pricing for energy while ensuring vehicle security. It also aims to guarantee unlinkability and anonymity for EVs, albeit based on an older version of DAA. However, the protocol requires an active connection between the CS, CSO, and the energy provider, which may not always be feasible in real-world scenarios. Furthermore, it lacks compatibility with established standards like ISO 15118, OCPP, or OICP.

In a distinct approach presented in [113], an anonymous payment system for EVs using the E-Cash technique was proposed, rather than introducing an anonymous authentication mechanism. This system enables the distribution of anonymous coins to EVs, which can be used for direct payment at the CS for consumed energy. Subsequently, the CS redeems the coins with the MO, which initially distributed them. However, this approach demands a unique architecture compared to established standards like ISO 15118 and entails considerable efforts to prevent double spending of coins.

Knirsch et al. introduced a blockchain-based protocol in [104, 186], enabling energy providers to offer smart contracts through a CS, allowing customers to select from various

---

offers. While the protocol claims to prioritize privacy, the authors acknowledge that customers' charging processes can be linked through customer IDs and public keys within the blockchain used for authentication. To mitigate this, the proposed solution involves changing the ID and authentication keys for every charging process. Nevertheless, this approach introduces fresh challenges, such as key distribution and management, which still need to be addressed in their work.

Kern et al. have extended the presented approach in [100] by the capability to handle CSs that operate offline, the reservation of CSs, and a process to install contract credentials automatically on EVs.

## **7.10. Summary on the Security of Charging Infrastructure**

This chapter demonstrates the potential for manipulating the power grid by exploiting vulnerabilities in EV charging processes. We have systematically examined a spectrum of attack vectors that could be exploited to disrupt the grid's stability. Furthermore, we have outlined attacker strategies and substantiated our findings through a practical assessment, wherein we executed multiple attacks on industry-standard protocols such as ISO 15118 and OCPP. These findings confirm our attack feasibility rating given in Chapter 4.

Moreover, our analysis of prevalent PnC protocols, including ISO 15118, OCPP, and OICP, has revealed an absence of privacy-preserving measures within these standards. Consequently, private data of charging and billing remain inadequately secured, leaving vulnerabilities such as the creation of user movement profiles unmitigated.

Building on our comprehensive analysis and considering the GDPR, we have defined stringent requirements for our proposed privacy-preserving solution aimed at rectifying these deficiencies. Our approach advocates the incorporation of a TPM within the vehicle, coupled with integrating a DAA scheme into existing PnC protocols.

An evaluation of our proposed approach underscores its ability to fulfill the defined privacy prerequisites while incurring minimal additional overhead to the existing protocols. In conclusion, our privacy extension for PnC protocols allows users to anonymously charge their EVs while concurrently affording service providers the means to securely bill their customers. Thus, the approach significantly reduces their risk of exposing private data in a practical and feasible manner.

---

## 8. Conclusion and Future Work

---

In this work, we have tackled the multifaceted domain of modern EV cybersecurity. Our work addresses some of the most critical aspects of EV security and resilience, aiming to pave the way for safer and more secure electric vehicles in an ever-evolving automotive landscape.

In the initial chapters, we established the foundational framework for our research by emphasizing the paramount importance of TARA in the context of EV cybersecurity. A successful TARA forms the bedrock for designing robust and resilient security mechanisms in modern electric vehicles. We introduced a novel approach to assess the attack surface, which includes asset identification, threat scenario analysis, attack path evaluation, and the assessment of attack feasibility. This method, developed in alignment with ISO/SAE 21434 standards, introduces automation to expedite the process, reduce errors, and enhance efficiency. Furthermore, our investigation into attack feasibility serves as a crucial tool within TARA, providing insights into potential threat scenarios and their likelihood. The automation of this process provides a faster and less error-prone alternative to manual attack derivation.

We used the process to examine high-impact risks, such as manipulating driving behavior, privacy breaches in the charging process, and threats to power grid stability through EV charging. Our research led us to identify paths with high or medium attack feasibility, paving the way for subsequent chapters to address these challenges. The following chapters delve into specific security aspects, including CAN bus security, TLS in Automotive Ethernet, and the security of the EV charging infrastructure.

In the chapter on CAN bus security (Chapter 5), we designed and developed the BusCount protocol as a secure communication solution to protect automotive networks. Our protocol demonstrated clear advantages over state-of-the-art solutions, particularly in mitigating message manipulation and replay attacks. These security properties are proven using the Tamarin prover. We discussed the potential implementation of BusCount into the firmware of a CAN controller, which could further reduce the cost of our solution. Future work may

---


also explore the integration of hardware security modules to safeguard cryptographic keys from potential attackers.

In the chapter on security in Automotive Ethernet (Chapter 6), we presented a communication strategy to secure Automotive Ethernet communication in vehicles, assuming the security assumptions of TLS hold. This strategy offers guarantees of authenticity, confidentiality, and integrity through encryption and authentication mechanisms. We also discussed the performance requirements for TLS in-vehicle networks, highlighting the feasibility of using typical automotive hardware for this purpose. Furthermore, we identified three distinct types of MitM attacks that can be executed on the Automotive Ethernet protocol SOME/IP using the formal verification tool Tamarin. The attacks even work when security protocols are in place. Our implementation and evaluation of these attacks on two real-world SOME/IP libraries have demonstrated their effectiveness. We propose two security extensions to counteract these attacks: Secure SOME/IP service discovery and session establishment using restricted certificates and digital signatures (SESO-RC) and Secure SOME/IP service discovery and session establishment using an authorization server (SESO-AS). We have conducted a Tamarin analysis to evaluate the security of these extensions, and our implementation supports their practical feasibility.

The chapter on the security of charging infrastructure Chapter 7 addressed the potential manipulation of the power grid through vulnerabilities in EV charging processes. We systematically examined various attack vectors, identified attacker strategies, and conducted practical assessments to validate our findings. We also uncovered a lack of privacy-preserving measures in prevalent PnC protocols, which could lead to the exposure of private data during charging and billing processes. Our proposed solution advocates the integration of TPMs within vehicles and incorporating DAA schemes into existing PnC protocols. This approach enhances user privacy while ensuring secure billing processes.

In summary, this doctoral thesis contributes to the advancement of EV cybersecurity by addressing pivotal aspects of risk assessment, CAN bus security, Automotive Ethernet security, and the security of EV charging infrastructure. Integrating automated threat and risk analysis, novel security protocols, and robust privacy measures aligns with the growing demand for secure, resilient electric vehicles. As the automotive landscape continues to evolve, these findings and solutions provide a solid foundation for creating a safer and more secure environment for electric cars.

As discussed, there are multiple topics in the context of automotive security to follow up research on in the future. The most prominent trend is autonomous driving, which leads to multiple safety problems in case of cyber attacks. This includes attacks on the AI



---

driving system, like image recognition, but also attacks against sensors. Furthermore, this thesis does not address the topics of ECU integrity and software vulnerabilities.



---

## Bibliography

---

- [1] Dare Abodunrin, Yoan Miche, and Silke Holtmanns. “Some dangers from 2G networks legacy support and a possible mitigation”. In: *2015 IEEE Conference on Communications and Network Security, CNS 2015, Florence, Italy, September 28-30, 2015*. IEEE, 2015, pp. 585–593. DOI: 10.1109/CNS.2015.7346872. URL: <https://doi.org/10.1109/CNS.2015.7346872>.
- [2] Samrat Acharya et al. “Cybersecurity of Smart Electric Vehicle Charging: A Power Grid Perspective”. In: *IEEE Access* 8 (2020), pp. 214434–214453. DOI: 10.1109/ACCESS.2020.3041074. URL: <https://doi.org/10.1109/ACCESS.2020.3041074>.
- [3] Cristina Alcaraz, Javier López, and Stephen D. Wolthusen. “OCPP Protocol: Security Threats and Challenges”. In: *IEEE Trans. Smart Grid* 8.5 (2017), pp. 2452–2459. DOI: 10.1109/TSG.2017.2669647. URL: <https://doi.org/10.1109/TSG.2017.2669647>.
- [4] Jean-Philippe Aumasson and Daniel J. Bernstein. “SipHash: A Fast Short-Input PRF”. In: *Progress in Cryptology - INDOCRYPT 2012, 13th International Conference on Cryptology in India, Kolkata, India, December 9-12, 2012. Proceedings*. Ed. by Steven D. Galbraith and Mridul Nandi. Vol. 7668. Lecture Notes in Computer Science. Springer, 2012, pp. 489–508. DOI: 10.1007/978-3-642-34931-7\_28. URL: [https://doi.org/10.1007/978-3-642-34931-7\\_28](https://doi.org/10.1007/978-3-642-34931-7_28).
- [5] CAN in Automation (CiA). *CAN XL specifications and test plans - Part 1: Data link layer and physical coding sub-layer requirements*. en. Draft Specification CiA 610-1. Nürenberg, Germany: CAN in Automation e. V., Mar. 2022. URL: <https://www.can-cia.org/groups/specifications/>.
- [6] AUTOSAR. “SOME/IP Protocol Specification”. In: vol. R21-11. Nov. 2021.
- [7] AUTOSAR. “SOME/IP Service Discovery Protocol Specification”. In: vol. R21-11. Nov. 2021.

- 
- 
- [8] AUTOSAR. “Specification of Secure Onboard Communication”. In: vol. R21-11. Nov. 2021.
- [9] Richard Baker and Ivan Martinovic. “Losing the Car Keys: Wireless PHY-Layer Insecurity in EV Charging”. In: *28th USENIX Security Symposium, USENIX Security 2019, Santa Clara, CA, USA, August 14-16, 2019*. Ed. by Nadia Heninger and Patrick Traynor. USENIX Association, 2019, pp. 407–424. URL: <https://www.usenix.org/conference/usenixsecurity19/presentation/baker>.
- [10] Kaibin Bao et al. “A threat analysis of the vehicle-to-grid charging protocol ISO 15118”. In: *Comput. Sci. Res. Dev.* 33.1-2 (2018), pp. 3–12. DOI: 10.1007/s00450-017-0342-y. URL: <https://doi.org/10.1007/s00450-017-0342-y>.
- [11] Elaine Barker. *Recommendation for Key Management*. Special Publication 800-57 Part 1 Rev. 5. May 2020. URL: <https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-57pt1r5.pdf>.
- [12] D. Basin et al. *Tamarin Prover*. 2020. URL: <https://tamarin-prover.github.io/>.
- [13] Giampaolo Bella et al. “TOUCAN: A proTocol tO secUre Controller Area Network”. In: *Proceedings of the ACM Workshop on Automotive Cybersecurity, AutoSec@CODASPY 2019, Richardson, TX, USA, March 27, 2019*. Ed. by Ziming Zhao, Qi Alfred Chen, and Gail-Joon Ahn. ACM, 2019, pp. 3–8. DOI: 10.1145/3309171.3309175. URL: <https://doi.org/10.1145/3309171.3309175>.
- [14] Richard Bellman. “On a routing problem”. In: *Quarterly of applied mathematics* 16.1 (1958), pp. 87–90.
- [15] Karthikeyan Bhargavan, Bruno Blanchet, and Nadim Kobeissi. “Verified Models and Reference Implementations for the TLS 1.3 Standard Candidate”. In: *2017 IEEE Symposium on Security and Privacy, SP 2017, San Jose, CA, USA, May 22-26, 2017*. IEEE Computer Society, 2017, pp. 483–502. DOI: 10.1109/SP.2017.26. URL: <https://doi.org/10.1109/SP.2017.26>.
- [16] P. Biondi and the Scapy community. *Scapy for python3 2.4.3rc1*. <https://github.com/secdev/scapy>. 2020. (Visited on 04/15/2019).
- [17] Evangelos Bitsikas and Christina Pöpper. “Don’t hand it Over: Vulnerabilities in the Handover Procedure of Cellular Telecommunications”. In: *ACSAC ’21: Annual Computer Security Applications Conference, Virtual Event, USA, December 6 - 10, 2021*. ACM, 2021, pp. 900–915. DOI: 10.1145/3485832.3485914. URL: <https://doi.org/10.1145/3485832.3485914>.



- 
- [18] BMW AG. *vsomeip 2.10.21*. <https://github.com/GENIVI/vsomeip>. 2018. (Visited on 04/26/2019).
- [19] Andrey Bogdanov et al. "PRESENT: An Ultra-Lightweight Block Cipher". In: *Cryptographic Hardware and Embedded Systems - CHES 2007, 9th International Workshop, Vienna, Austria, September 10-13, 2007, Proceedings*. Ed. by Pascal Paillier and Ingrid Verbauwhede. Vol. 4727. Lecture Notes in Computer Science. Springer, 2007, pp. 450–466. DOI: 10.1007/978-3-540-74735-2\_31. URL: [https://doi.org/10.1007/978-3-540-74735-2%5C\\_31](https://doi.org/10.1007/978-3-540-74735-2%5C_31).
- [20] Anastasia Bolovinou et al. "TARA+: Controllability-aware Threat Analysis and Risk Assessment for L3 Automated Driving Systems". In: *2019 IEEE Intelligent Vehicles Symposium, IV 2019, Paris, France, June 9-12, 2019*. IEEE, 2019, pp. 8–13. DOI: 10.1109/IVS.2019.8813999. URL: <https://doi.org/10.1109/IVS.2019.8813999>.
- [21] Julia Borghoff et al. "PRINCE - A Low-Latency Block Cipher for Pervasive Computing Applications - Extended Abstract". In: *Advances in Cryptology - ASIACRYPT 2012 - 18th International Conference on the Theory and Application of Cryptology and Information Security, Beijing, China, December 2-6, 2012. Proceedings*. Ed. by Xiaoyun Wang and Kazue Sako. Vol. 7658. Lecture Notes in Computer Science. Springer, 2012, pp. 208–225. DOI: 10.1007/978-3-642-34961-4\_14. URL: [https://doi.org/10.1007/978-3-642-34961-4%5C\\_14](https://doi.org/10.1007/978-3-642-34961-4%5C_14).
- [22] Aymen Boudguiga et al. "RACE: Risk analysis for cooperative engines". In: *7th International Conference on New Technologies, Mobility and Security, NTMS 2015, Paris, France, July 27-29, 2015*. Ed. by Mohamad Badra, Azzedine Boukerche, and Pascal Urien. IEEE, 2015, pp. 1–5. DOI: 10.1109/NTMS.2015.7266516. URL: <https://doi.org/10.1109/NTMS.2015.7266516>.
- [23] Ernest F. Brickell, Jan Camenisch, and Liqun Chen. "Direct anonymous attestation". In: *Proceedings of the 11th ACM Conference on Computer and Communications Security, CCS 2004, Washington, DC, USA, October 25-29, 2004*. Ed. by Vijayalakshmi Atluri, Birgit Pfitzmann, and Patrick D. McDaniel. ACM, 2004, pp. 132–145. DOI: 10.1145/1030083.1030103. URL: <https://doi.org/10.1145/1030083.1030103>.
- [24] Alessandro Bruni et al. "Formal Security Analysis of the MaCAN Protocol". In: *Integrated Formal Methods - 11th International Conference, IFM 2014, Bertinoro, Italy, September 9-11, 2014, Proceedings*. Ed. by Elvira Albert and Emil Sekerinski. Vol. 8739. Lecture Notes in Computer Science. Springer, 2014, pp. 241–255. DOI:

- 
- 10.1007/978-3-319-10181-1\_15. URL: [https://doi.org/10.1007/978-3-319-10181-1\\_15](https://doi.org/10.1007/978-3-319-10181-1%5C_15).
- [25] Jo Van Bulck, Jan Tobias Mühlberg, and Frank Piessens. “VulCAN: Efficient Component Authentication and Software Isolation for Automotive Control Networks”. In: *Proceedings of the 33rd Annual Computer Security Applications Conference, Orlando, FL, USA, December 4-8, 2017*. ACM, 2017, pp. 225–237. DOI: 10.1145/3134600.3134623. URL: <https://doi.org/10.1145/3134600.3134623>.
- [26] Cornelius Butzkamm and Konstantin Brand. “E/E Architecture in the HARRI Innovation Platform”. In: *ATZelectronics worldwide* 15.3 (Mar. 2020). URL: <https://doi.org/10.1007/s38314-019-0160-z>.
- [27] Zhiqiang Cai et al. “0-days & Mitigations: Roadways to Exploit and Secure Connected BMW Cars”. In: *Black Hat USA 2019* (2019), p. 39.
- [28] Jan Camenisch et al. “One TPM to Bind Them All: Fixing TPM 2.0 for Provably Secure Anonymous Attestation”. In: *2017 IEEE Symposium on Security and Privacy, SP 2017, San Jose, CA, USA, May 22-26, 2017*. IEEE Computer Society, 2017, pp. 901–920. DOI: 10.1109/SP.2017.22. URL: <https://doi.org/10.1109/SP.2017.22>.
- [29] Yulong Cao et al. “Adversarial Sensor Attack on LiDAR-based Perception in Autonomous Driving”. In: *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security, CCS 2019, London, UK, November 11-15, 2019*. Ed. by Lorenzo Cavallaro et al. ACM, 2019, pp. 2267–2281. DOI: 10.1145/3319535.3339815. URL: <https://doi.org/10.1145/3319535.3339815>.
- [30] Angelo Caposelle et al. “Security as a CoAP resource: An optimized DTLS implementation for the IoT”. In: *2015 IEEE International Conference on Communications, ICC 2015, London, United Kingdom, June 8-12, 2015*. IEEE, 2015, pp. 549–554. DOI: 10.1109/ICC.2015.7248379. URL: <https://doi.org/10.1109/ICC.2015.7248379>.
- [31] Stephen Checkoway et al. “Comprehensive Experimental Analyses of Automotive Attack Surfaces”. In: *20th USENIX Security Symposium, San Francisco, CA, USA, August 8-12, 2011, Proceedings*. USENIX Association, 2011. URL: [http://static.usenix.org/events/sec11/tech/full%5C\\_papers/Checkoway.pdf](http://static.usenix.org/events/sec11/tech/full%5C_papers/Checkoway.pdf).
- [32] Huajiang Chen and Wu Ming. *Remotely Rooting Charging Station For Fun And Maybe Profit*. Aug. 2021. URL: <http://chv.link/kevin2600>.

- 
- [33] Ju-Ho Choi, Sung-Gi Min, and Youn-Hee Han. “MACsec Extension over Software-Defined Networks for in-Vehicle Secure Communication”. In: *Tenth International Conference on Ubiquitous and Future Networks, ICUFN 2018, Prague, Czech Republic, July 3-6, 2018*. IEEE, 2018, pp. 180–185. DOI: 10.1109/ICUFN.2018.8436963. URL: <https://doi.org/10.1109/ICUFN.2018.8436963>.
- [34] SeDaFa Project Consortium. *Anforderungsanalyse für Selbstschutz im vernetzten Fahrzeug*. Aug. 2017. URL: [https://sedafa-projekt.de/media/D1\\_final.pdf](https://sedafa-projekt.de/media/D1_final.pdf).
- [35] SeDaFa Project Consortium. *Angreifermodell für Selbstschutz im vernetzten Fahrzeug*. Aug. 2017. URL: [https://sedafa-projekt.de/media/D1.5\\_final.pdf](https://sedafa-projekt.de/media/D1.5_final.pdf).
- [36] Broadcom Corporation. *BroadR-Reach® Physical Layer Transceiver Specification For Automotive Applications*. May 2014.
- [37] Gianpiero Costantino and Ilaria Matteucci. “Reversing Kia Motors head unit to discover and exploit software vulnerabilities”. In: *Journal of Computer Virology and Hacking Techniques* (May 2022). DOI: 10.1007/s11416-022-00430-5.
- [38] Cas Cremers et al. “A Comprehensive Symbolic Analysis of TLS 1.3”. In: *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security, CCS 2017, Dallas, TX, USA, October 30 - November 03, 2017*. Ed. by Bhavani M. Thuraisingham et al. ACM, 2017, pp. 1773–1788. DOI: 10.1145/3133956.3134063. URL: <https://doi.org/10.1145/3133956.3134063>.
- [39] Adrian Dabrowski, Johanna Ullrich, and Edgar R. Weippl. “Grid Shock: Coordinated Load-Changing Attacks on Power Grids: The Non-Smart Power Grid is Vulnerable to Cyber Attacks as Well”. In: *Proceedings of the 33rd Annual Computer Security Applications Conference, Orlando, FL, USA, December 4-8, 2017*. ACM, 2017, pp. 303–314. DOI: 10.1145/3134600.3134639. URL: <https://doi.org/10.1145/3134600.3134639>.
- [40] M. Dalheimer. *Chaos Computer Club hacks e-motor charging stations*. <https://www.ccc.de/en/updates/2017/e-motor>. Last visited on 26/06/2018. Dec. 2017.
- [41] Udo Dannebaum, Harald Zweck, and Karsten Schmidt. “Gbit Ethernet - The Solution for Future In-Vehicle Network Requirements?” In: *SAE International Journal of Passenger Cars - Electronic and Electrical Systems* 8.2 (Apr. 2015), pp. 289–295. ISSN: 1946-4614. DOI: <https://doi.org/10.4271/2015-01-0200>. URL: <https://doi.org/10.4271/2015-01-0200>.

- 
- [42] Yuri Gil Dantas, Vivek Nigam, and Harald Ruess. “Security Engineering for ISO 21434”. In: *CoRR* abs/2012.15080 (2020). arXiv: 2012.15080. URL: <https://arxiv.org/abs/2012.15080>.
- [43] Marco De Vincenzi et al. “A Systematic Review on Security Attacks and Countermeasures in Automotive Ethernet”. In: *ACM Comput. Surv.* 56.6 (Jan. 2024). ISSN: 0360-0300. DOI: 10.1145/3637059. URL: <https://doi.org/10.1145/3637059>.
- [44] Mahdi Dibaei et al. “Attacks and defences on intelligent connected vehicles: a survey”. In: *Digit. Commun. Networks* 6.4 (2020), pp. 399–421. DOI: 10.1016/j.dcan.2020.04.007. URL: <https://doi.org/10.1016/j.dcan.2020.04.007>.
- [45] Tim Dierks and Eric Rescorla. *The Transport Layer Security (TLS) Protocol Version 1.2*. RFC 5246 (Proposed Standard). Updated by RFCs 5746, 5878, 6176. Internet Engineering Task Force, Aug. 2008. URL: <http://www.ietf.org/rfc/rfc5246.txt>.
- [46] Christoph Dobraunig, Florian Mendel, and Martin Schl affer. “Differential Cryptanalysis of SipHash”. In: *Selected Areas in Cryptography - SAC 2014 - 21st International Conference, Montreal, QC, Canada, August 14-15, 2014, Revised Selected Papers*. Ed. by Antoine Joux and Amr M. Youssef. Vol. 8781. Lecture Notes in Computer Science. Springer, 2014, pp. 165–182. DOI: 10.1007/978-3-319-13051-4\_10. URL: [https://doi.org/10.1007/978-3-319-13051-4\\_10](https://doi.org/10.1007/978-3-319-13051-4_10).
- [47] Danny Dolev and Andrew Chi-Chih Yao. “On the security of public key protocols”. In: *IEEE Trans. Inf. Theory* 29.2 (1983), pp. 198–207. DOI: 10.1109/TIT.1983.1056650. URL: <https://doi.org/10.1109/TIT.1983.1056650>.
- [48] S bastien Dudek, Jean-Christophe Delaunay, and Vincent Fargues. “V2G Injector: Whispering to cars and charging units through the Power-Line”. In: *Proceedings of the SSTIC (Symposium sur la s curit  des technologies de l’information et des communications), Rennes, France. 2019*, pp. 5–7. URL: [https://www.sstic.org/media/SSTIC2019/SSTIC-actes/v2g\\_injector\\_playing\\_with\\_electric\\_cars\\_and\\_chargi/SSTIC2019-Article-v2g\\_injector\\_playing\\_with\\_electric\\_cars\\_and\\_charging\\_stations\\_via\\_powerline-dudek.pdf](https://www.sstic.org/media/SSTIC2019/SSTIC-actes/v2g_injector_playing_with_electric_cars_and_chargi/SSTIC2019-Article-v2g_injector_playing_with_electric_cars_and_charging_stations_via_powerline-dudek.pdf).
- [49] Michael D ll et al. “High-speed Curve25519 on 8-bit, 16-bit, and 32-bit microcontrollers”. In: *Des. Codes Cryptogr.* 77.2-3 (2015), pp. 493–514. DOI: 10.1007/s10623-015-0087-1. URL: <https://doi.org/10.1007/s10623-015-0087-1>.

- 
- [50] Charging Interface Initiative e.V. *CharIN Implementation Guide to Plug and Charge in the context of ISO 15118*. Mar. 2022.
- [51] ETSI. *ETSI TS 102 165-1 V5.2.3 (2017-10) – CYBER; Methods and protocols; Part 1: Method and pro forma for Threat, Vulnerability, Risk Analysis (TVRA)*. Tech. rep. ETSI, 2017.
- [52] European Telecommunications Standards Institute (ETSI). *Intelligent Transport Systems (ITS); V2X Applications; Part 1: Road Hazard Signalling (RHS) Application Requirements Specification*. en. Standard Standard ETSI TS 101 539-1. Sophia Antipolis, France: European Telecommunications Standards Institute, Aug. 2013. URL: [www.etsi.org/deliver/etsi\\_ts/101500\\_101599/10153901/01.01.01\\_60/ts\\_10153901v010101p.pdf](http://www.etsi.org/deliver/etsi_ts/101500_101599/10153901/01.01.01_60/ts_10153901v010101p.pdf).
- [53] “General Data Protection Regulation”. In: *Official Journal of the European Union* L119 (May 2016), pp. 1–88.
- [54] R. Falk and S. Fries. “Electric vehicle charging infrastructure security considerations and approaches”. In: *INTERNET 2012: The Fourth International Conference on Evolving Internet* (June 2012), pp. 58–64. URL: <http://www.fi-ppp-finsen.y.eu/wp-content/uploads/2013/05/Electric-Vehicle-Charging-Infrastructure-%E2%80%93-Security-Considerations-and-Approaches.pdf>.
- [55] Marouane Fazouane et al. “Formal Verification of Privacy Properties in Electric Vehicle Charging”. In: *Engineering Secure Software and Systems - 7th International Symposium, ESSoS 2015, Milan, Italy, March 4-6, 2015. Proceedings*. Ed. by Frank Piessens, Juan Caballero, and Nataliia Bielova. Vol. 8978. Lecture Notes in Computer Science. Springer, 2015, pp. 17–33. DOI: 10.1007/978-3-319-15618-7\_2. URL: [https://doi.org/10.1007/978-3-319-15618-7%5C\\_2](https://doi.org/10.1007/978-3-319-15618-7%5C_2).
- [56] Loïc Fejoz, Jan Seyler, and Nicolas Navet. “Insights on the Configuration and Performances of SOME/IP Service Discovery”. In: *SAE International Journal of Passenger Cars - Electronic and Electrical Systems* 8.1 (Apr. 2015), pp. 124–129. ISSN: 1946-4614. DOI: <https://doi.org/10.4271/2015-01-0197>. URL: <https://doi.org/10.4271/2015-01-0197>.
- [57] Marc Fischlin and Felix Günther. “Replay Attacks on Zero Round-Trip Time: The Case of the TLS 1.3 Handshake Candidates”. In: *2017 IEEE European Symposium on Security and Privacy, EuroS&P 2017, Paris, France, April 26-28, 2017*. IEEE, 2017, pp. 60–75. DOI: 10.1109/EuroSP.2017.18. URL: <https://doi.org/10.1109/EuroSP.2017.18>.

- 
- [58] L. R. Ford. *Network Flow Theory*. Santa Monica, CA: RAND Corporation, 1956.
- [59] Ian D. Foster et al. “Fast and Vulnerable: A Story of Telematic Failures”. In: *9th USENIX Workshop on Offensive Technologies, WOOT '15, Washington, DC, USA, August 10-11, 2015*. Ed. by Aurélien Francillon and Thomas Ptacek. USENIX Association, 2015. URL: <https://www.usenix.org/conference/woot15/workshop-program/presentation/foster>.
- [60] Andreas Fuchs, Christoph Krauß, and Jürgen Repp. “Advanced Remote Firmware Upgrades Using TPM 2.0”. In: *ICT Systems Security and Privacy Protection - 31st IFIP TC 11 International Conference, SEC 2016, Ghent, Belgium, May 30 - June 1, 2016, Proceedings*. Ed. by Jaap-Henk Hoepman and Stefan Katzenbeisser. Vol. 471. IFIP Advances in Information and Communication Technology. Springer, 2016, pp. 276–289. DOI: 10.1007/978-3-319-33630-5\_19. URL: [https://doi.org/10.1007/978-3-319-33630-5\\_19](https://doi.org/10.1007/978-3-319-33630-5_19).
- [61] *Global EV Outlook 2022*. France, Paris, 2020. URL: <https://www.iea.org/reports/global-ev-outlook-2022>.
- [62] Robert Bosch GmbH. *CAN with Flexible Data-Rate*. Apr. 2012.
- [63] Vector Informatik GmbH. *CANoe*. 2019. URL: <https://www.vector.com/int/en/products/products-a-z/software/canoe/> (visited on 03/20/2020).
- [64] Andre Groll and Christoph Ruland. “Secure and Authentic Communication on Existing In-Vehicle Networks”. In: *2009 IEEE Intelligent Vehicles Symposium*. July 2009, pp. 1093–1097. DOI: 10.1109/IVS.2009.5164434.
- [65] Bogdan Groza, Lucian Popa, and Pal-Stefan Murvay. “Highly Efficient Authentication for CAN by Identifier Reallocation With Ordered CMACs”. In: *IEEE Trans. Veh. Technol.* 69.6 (2020), pp. 6129–6140. DOI: 10.1109/TVT.2020.2990954. URL: <https://doi.org/10.1109/TVT.2020.2990954>.
- [66] Bogdan Groza et al. “LiBrA-CAN: A Lightweight Broadcast Authentication Protocol for Controller Area Networks”. In: *Cryptology and Network Security, 11th International Conference, CANS 2012, Darmstadt, Germany, December 12-14, 2012. Proceedings*. Ed. by Josef Pieprzyk, Ahmad-Reza Sadeghi, and Mark Manulis. Vol. 7712. Springer, 2012, pp. 185–200. DOI: 10.1007/978-3-642-35404-5\_15. URL: [https://doi.org/10.1007/978-3-642-35404-5\\_15](https://doi.org/10.1007/978-3-642-35404-5_15).

- 
- [67] Sigrid Grgens and Daniel Zelle. "A Hardware Based Solution for Freshness of Secure Onboard Communication in Vehicles". In: *Computer Security - ESORICS 2018 International Workshops, CyberICPS 2018 and SECPRE 2018, Barcelona, Spain, September 6-7, 2018, Revised Selected Papers*. Ed. by Sokratis K. Katsikas et al. Vol. 11387. Lecture Notes in Computer Science. Springer, 2018, pp. 53–68. ISBN: 978-3-030-12785-5. DOI: 10.1007/978-3-030-12786-2\_4. URL: [https://doi.org/10.1007/978-3-030-12786-2%5C\\_4](https://doi.org/10.1007/978-3-030-12786-2%5C_4).
- [68] Piroska Haller et al. "VetaDetect: Vehicle tampering detection with closed-loop model ensemble". In: *International Journal of Critical Infrastructure Protection* 37 (2022), p. 100525. ISSN: 1874-5482. DOI: <https://doi.org/10.1016/j.ijcip.2022.100525>. URL: <https://www.sciencedirect.com/science/article/pii/S1874548222000154>.
- [69] Peter Hank et al. "Automotive ethernet: in-vehicle networking and smart mobility". In: *Design, Automation and Test in Europe, DATE 13, Grenoble, France, March 18-22, 2013*. Ed. by Enrico Macii. EDA Consortium San Jose, CA, USA / ACM DL, 2013, pp. 1735–1739. DOI: 10.7873/DATE.2013.349. URL: <https://doi.org/10.7873/DATE.2013.349>.
- [70] Marit Hansen, Meiko Jensen, and Martin Rost. "Protection Goals for Privacy Engineering". In: *2015 IEEE Symposium on Security and Privacy Workshops, SPW 2015, San Jose, CA, USA, May 21-22, 2015*. IEEE Computer Society, 2015, pp. 159–166. DOI: 10.1109/SPW.2015.13. URL: <https://doi.org/10.1109/SPW.2015.13>.
- [71] Oliver Hartkopp, Cornel Reuber, and Roland Schilling. "MaCAN - Message Authenticated CAN". In: *10th Embedded Security in Cars (escar) Conference Europe 2012*. Vol. 10. International School of IT Security AG. Berlin, Germany, Nov. 2012.
- [72] Simon Haverkamp and Martin Simons. *Cybersecurity in the charging ecosystem. Status quo and stakeholder ambitions*. Tech. rep. umlaut energy GmbH on behalf of Mercedes-Benz AG, Feb. 2022. URL: [https://www.charin.global/media/pages/news/charin-task-force-cybersecurity/af8d69da69-1645626738/20220203-charging-ecosystem-stakeholder-landscape-rev\\_1.3.pdf](https://www.charin.global/media/pages/news/charin-task-force-cybersecurity/af8d69da69-1645626738/20220203-charging-ecosystem-stakeholder-landscape-rev_1.3.pdf).
- [73] Jeremias Hawran. *Harmon-E - interaktive Systemarchitektur*. 2022. URL: <https://sysarc.ffe.de/>.

- 
- [74] Ahmed Hazem and Hossam Mahmoud Ahmad Fahmy. “LCAP - A Lightweight CAN Authentication Protocol for Security In-Vehicle-Networks”. In: *10th Embedded Security in Cars (escar) Conference Europe 2012*. Vol. 10. International School of IT Security AG. Berlin, Germany, Nov. 2012.
- [75] HEAVENS consortium. *HEAVENS - HEALing Vulnerabilities to Enhance Software Security and Safety*. <https://research.chalmers.se/en/project/5809>. last accessed 2020-01-13. 2016.
- [76] Nadine Herold et al. “Anomaly detection for SOME/IP using complex event processing”. In: *2016 IEEE/IFIP Network Operations and Management Symposium, NOMS 2016, Istanbul, Turkey, April 25-29, 2016*. Ed. by Sema Oktug et al. IEEE, 2016, pp. 1221–1226. DOI: 10.1109/NOMS.2016.7502991. URL: <https://doi.org/10.1109/NOMS.2016.7502991>.
- [77] Christina Höfer et al. “POPCORN: privacy-preserving charging for emobility”. In: *CyCAR’13, Proceedings of the 2013 ACM Workshop on Security, Privacy and Dependability for CyberVehicles, Co-located with CCS 2013, November 4, 2013, Berlin, Germany*. Ed. by Ahmad-Reza Sadeghi, Cliff Wang, and Hervé Seudie. ACM, 2013, pp. 37–48. DOI: 10.1145/2517968.2517971. URL: <https://doi.org/10.1145/2517968.2517971>.
- [78] Syed Rafiul Hussain et al. “LTEInspector: A Systematic Approach for Adversarial Testing of 4G LTE”. In: *25th Annual Network and Distributed System Security Symposium, NDSS 2018, San Diego, California, USA, February 18-21, 2018*. The Internet Society, 2018. URL: [https://www.ndss-symposium.org/wp-content/uploads/2018/02/ndss2018%5C\\_02A-3%5C\\_Hussain%5C\\_paper.pdf](https://www.ndss-symposium.org/wp-content/uploads/2018/02/ndss2018%5C_02A-3%5C_Hussain%5C_paper.pdf).
- [79] IEEE Computer Society. *Media Access Control (MAC) Security*. IEEE Std 802.1AE-2006. 2006.
- [80] IEEE Computer Society. *Media Access Control (MAC) Security Amendment 1: GCM-AES-256 Cipher Suite*. IEEE Std 802.1AEbn-2011. 2011.
- [81] “IEEE Standard for Ethernet Amendment 1: Physical Layer Specifications and Management Parameters for 100 Mb/s Operation over a Single Balanced Twisted Pair Cable (100BASE-T1)”. In: *IEEE Std 802.3bw-2015 (Amendment to IEEE Std 802.3-2015)* (2016), pp. 1–88. DOI: 10.1109/IEEESTD.2016.7433918.
- [82] *Improved security for OCPP 1.6-J*. 3rd ed. Open Charge Alliance. Feb. 2022. URL: <https://www.openchargealliance.org/uploads/files/OCPP-1.6-security-whitepaper-edition-3.zip>.



- 
- [83] Infineon Technologies. *Automotive Security - Trusted Driving*. <https://www.infineon.com/cms/en/applications/automotive/automotive-security/>. Last visited on 30/09/2023. 2017.
- [84] Marco Iorio et al. "Protecting In-Vehicle Services: Security-Enabled SOME/IP Middleware". In: *IEEE Veh. Technol. Mag.* 15.3 (2020), pp. 77–85. DOI: 10.1109/MVT.2020.2980444. URL: <https://doi.org/10.1109/MVT.2020.2980444>.
- [85] ISO Central Secretary. *Information technology — Trusted platform module library*. en. Standard ISO 11898-1:2015. Geneva, CH: International Organization for Standardization, Dec. 2015. URL: <https://www.iso.org/standard/63648.html>.
- [86] ISO Central Secretary. *Road vehicles – Controller area network (CAN)*. en. Standard ISO 11898-1:2015. Geneva, CH: International Organization for Standardization, Dec. 2015. URL: <https://www.iso.org/standard/63648.html>.
- [87] ISO/IEC JTC 1/SC 27 Information security, cybersecurity and privacy protection. *ISO/IEC 18045:2022 Information security, cybersecurity and privacy protection - Evaluation criteria for IT security - Methodology for IT security evaluation*. en. Standard ISO/IEC 18045:2022. Geneva, CH, Aug. 2022. URL: <https://www.iso.org/standard/72889.html>.
- [88] ISO/TC 22/SC 31 Data communication. *Road vehicles - Vehicle to grid communication interface - Part 20: Network and application protocol requirements*. en. Standard ISO 15118-20:2022. Geneva, CH, Apr. 2022. URL: <https://www.iso.org/standard/77845.html>.
- [89] ISO/TC 22/SC 31 Data communication. "Road vehicles – Vehicle to grid communication interface – Part 1: General information and use-case definition". en. In: ISO 15118-1:2019 (Apr. 2013). URL: <https://www.iso.org/standard/69113.html>.
- [90] ISO/TC 22/SC 31 Data communication. *Road vehicles – Vehicle to grid communication interface – Part 2: Network and application protocol requirements*. en. Standard ISO 15118-2:2014. Geneva, CH, Apr. 2014. URL: <https://www.iso.org/standard/55366.html>.
- [91] ISO/TC 22/SC 31 Data communication. *Road vehicles – Vehicle to grid communication interface – Part 3: Physical and data link layer requirements*. en. Standard ISO 15118-3:2015. Geneva, CH, May 2015. URL: <https://www.iso.org/standard/59675.html>.

- 
- [92] ISO/TC 22/SC 31 Data communication. *Road vehicles — Diagnostic communication over Internet Protocol (DoIP) — Part 2: Transport protocol and network layer services*. en. Standard ISO 13400-2:2019. Geneva, CH: International Organization for Standardization, Dec. 2019. URL: <https://www.iso.org/standard/71044.html>.
- [93] ISO/TC 22/SC 32 Electrical and electronic components and general system aspects. *ISO 24089:2023 Road vehicles — Software update engineering*. en. Standard ISO 24089:2023. Geneva, CH, Feb. 2023. URL: <https://www.iso.org/standard/77796.html>.
- [94] ISO/TC 22/SC 32 Electrical and electronic components and general system aspects. *ISO/PAS 5112:2022 Road vehicles — Guidelines for auditing cybersecurity engineering*. en. Standard ISO/PAS 5112:2022. Geneva, CH, Mar. 2022. URL: <https://www.iso.org/standard/77796.html>.
- [95] ISO/TC 22/SC 32 Electrical and electronic components and general system aspects. *ISO/SAE 21434:2021 Road vehicles — Cybersecurity engineering*. en. Standard ISO/SAE 21434:2021. Geneva, CH, Aug. 2021. URL: <https://www.iso.org/standard/70918.html>.
- [96] Mathias Johanson, Pål Dahle, and Andreas Soderberg. “Remote Vehicle Diagnostics over the Internet using the DoIP Protocol”. In: *The Sixth International Conference on Systems and Networks Communications*. 2011.
- [97] Skyler Johnson et al. “Human and Organizational Factors in Public Key Certificate Authority Failures”. In: *CCS '21: 2021 ACM SIGSAC Conference on Computer and Communications Security, Virtual Event, Republic of Korea, November 15 - 19, 2021*. Ed. by Yongdae Kim et al. ACM, 2021, pp. 2414–2416. DOI: 10.1145/3460120.3485360. URL: <https://doi.org/10.1145/3460120.3485360>.
- [98] Sye Loong Keoh et al. *DTLS-based Multicast Security in Constrained Environments*. Internet-Draft draft-keoh-dice-multicast-security-08. Work in Progress. Internet Engineering Task Force, July 2014. 22 pp. URL: <https://tools.ietf.org/html/draft-keoh-dice-multicast-security-08>.
- [99] Dustin Kern and Christoph Krauß. “Analysis of E-Mobility-based Threats to Power Grid Resilience”. In: *CSCS '21: Computer Science in Cars Symposium, Ingolstadt, Germany, 30 November 2021*. Ed. by Björn Brücher et al. ACM, 2021, 10:1–10:12. DOI: 10.1145/3488904.3493385. URL: <https://doi.org/10.1145/3488904.3493385>.

- 
- [100] Dustin Kern, Timm Lauser, and Christoph Krauß. “Integrating Privacy into the Electric Vehicle Charging Architecture”. In: *Proc. Priv. Enhancing Technol.* 2022.3 (2022), pp. 140–158. DOI: 10.56553/POPETS-2022-0066. URL: <https://doi.org/10.56553/popets-2022-0066>.
- [101] D Keuper and T Alkemade. “The connected car ways to get unauthorized access and potential implications”. In: *Computest, Zoetermeer, The Netherlands, Tech. Rep* (2018).
- [102] Shawal Khan et al. “Security Challenges of Location Privacy in VANETs and State-of-the-Art Solutions: A Survey”. In: *Future Internet* 13.4 (2021), p. 96. DOI: 10.3390/fi13040096. URL: <https://doi.org/10.3390/fi13040096>.
- [103] Timo Kiravuo, Mikko Särelä, and Jukka Manner. “A Survey of Ethernet LAN Security”. In: *IEEE Commun. Surv. Tutorials* 15.3 (2013), pp. 1477–1491. DOI: 10.1109/SURV.2012.121112.00190. URL: <https://doi.org/10.1109/SURV.2012.121112.00190>.
- [104] Fabian Knirsch, Andreas Unterweger, and Dominik Engel. “Privacy-preserving blockchain-based electric vehicle charging with dynamic tariff decisions”. In: *Comput. Sci. Res. Dev.* 33.1-2 (2018), pp. 71–79. DOI: 10.1007/s00450-017-0348-5. URL: <https://doi.org/10.1007/s00450-017-0348-5>.
- [105] Sebastian Köhler et al. “Brokenwire : Wireless Disruption of CCS Electric Vehicle Charging”. In: *30th Annual Network and Distributed System Security Symposium, NDSS 2023, San Diego, California, USA, February 27 - March 3, 2023*. The Internet Society, 2023. URL: <https://www.ndss-symposium.org/ndss-paper/brokenwire-wireless-disruption-of-ccs-electric-vehicle-charging/>.
- [106] Karl Koscher et al. “Experimental Security Analysis of a Modern Automobile”. In: *31st IEEE Symposium on Security and Privacy, S&P 2010, 16-19 May 2010, Berkeley/Oakland, California, USA*. IEEE Computer Society, 2010, pp. 447–462. DOI: 10.1109/SP.2010.34. URL: <https://doi.org/10.1109/SP.2010.34>.
- [107] Todd Kosloff et al. “SS7 Messaging Attacks on Public Telephone Networks: Attack Scenarios and Detection”. In: *Workshop on the Scientific Aspects of Cyber Terrorism*. Washington, DC: ACM, Nov. 2002.
- [108] J. Kreissl. “Absicherung der SOME/IP Kommunikation bei Adaptive AUTOSAR”. MA thesis. Universität Stuttgart, 2017.

- 
- [109] Ryo Kurachi et al. “CaCAN - Centralized Authentication System in CAN”. In: *12th Embedded Security in Cars (escar) Conference Europe 2014*. Vol. 12. International School of IT Security AG. Hamburg, Germany, Nov. 2014.
- [110] Timm Lauser, Daniel Zelle, and Christoph Krauß. “Security Analysis of Automotive Protocols”. In: *CSCS '20: Computer Science in Cars Symposium, Feldkirchen, Germany, December 2, 2020*. Ed. by Björn Brücher et al. ACM, 2020, 11:1–11:12. DOI: 10.1145/3385958.3430482. URL: <https://doi.org/10.1145/3385958.3430482>.
- [111] Aljoscha Lautenbach and Mafijul Islam. *HEAVENS - HEALing Vulnerabilities to Enhance Software Security and Safety*. [http://autosec.se/wp-content/uploads/2018/03/HEAVENS\\_D2\\_v2.0.pdf](http://autosec.se/wp-content/uploads/2018/03/HEAVENS_D2_v2.0.pdf). last accessed 2020-01-13. Mar. 2016.
- [112] Youngwoo Lee and KyoungSoo Park. “Meeting the real-time constraints with standard Ethernet in an in-vehicle network”. In: *2013 IEEE Intelligent Vehicles Symposium (IV), Gold Coast City, Australia, June 23-26, 2013*. IEEE, 2013, pp. 1313–1318. DOI: 10.1109/IVS.2013.6629648. URL: <https://doi.org/10.1109/IVS.2013.6629648>.
- [113] Chao Li. “Anonymous Payment Mechanisms for Electric Car Infrastructure”. In: *Master's thesis* (2011).
- [114] Hongyang Li, György Dán, and Klara Nahrstedt. “Portunes+: Privacy-Preserving Fast Authentication for Dynamic Electric Vehicle Charging”. In: *IEEE Trans. Smart Grid* 8.5 (2017), pp. 2305–2313. DOI: 10.1109/TSG.2016.2522379. URL: <https://doi.org/10.1109/TSG.2016.2522379>.
- [115] Li Li et al. “Symbolic Analysis of an Electric Vehicle Charging Protocol”. In: *2014 19th International Conference on Engineering of Complex Computer Systems, Tianjin, China, August 4-7, 2014*. IEEE Computer Society, 2014, pp. 11–18. DOI: 10.1109/ICECCS.2014.11. URL: <https://doi.org/10.1109/ICECCS.2014.11>.
- [116] Xu Li et al. “Securing smart grid: cyber attacks, countermeasures, and challenges”. In: *IEEE Commun. Mag.* 50.8 (2012), pp. 38–45. DOI: 10.1109/MCOM.2012.6257525. URL: <https://doi.org/10.1109/MCOM.2012.6257525>.
- [117] Hyung-Taek Lim, Kay Weckemann, and Daniel Herrscher. “Performance Study of an In-Car Switched Ethernet Network without Prioritization”. In: *Communication Technologies for Vehicles - Third International Workshop, Nets4Cars/Nets4Trains 2011, Oberpfaffenhofen, Germany, March 23-24, 2011. Proceedings*. Ed. by Thomas Strang et al. Vol. 6596. Lecture Notes in Computer Science. Springer, 2011,

- 
- pp. 165–175. DOI: 10.1007/978-3-642-19786-4\_15. URL: [https://doi.org/10.1007/978-3-642-19786-4\\_15](https://doi.org/10.1007/978-3-642-19786-4_15).
- [118] Chung-Wei Lin and Alberto L. Sangiovanni-Vincentelli. “Cyber-Security for the Controller Area Network (CAN) Communication Protocol”. In: *2012 ASE International Conference on Cyber Security, Alexandria, VA, USA, December 14-16, 2012*. IEEE Computer Society, 2012, pp. 1–7. DOI: 10.1109/CyberSecurity.2012.7. URL: <https://doi.org/10.1109/CyberSecurity.2012.7>.
- [119] Gavin Lowe. “A Hierarchy of Authentication Specification”. In: *10th Computer Security Foundations Workshop (CSFW '97), June 10-12, 1997, Rockport, Massachusetts, USA*. IEEE Computer Society, 1997, pp. 31–44. DOI: 10.1109/CSFW.1997.596782. URL: <https://doi.org/10.1109/CSFW.1997.596782>.
- [120] Georg Macher et al. “A Review of Threat Analysis and Risk Assessment Methods in the Automotive Context”. In: *Computer Safety, Reliability, and Security - 35th International Conference, SAFECOMP 2016, Trondheim, Norway, September 21-23, 2016, Proceedings*. Ed. by Amund Skavhaug, Jérémie Guiochet, and Friedemann Bitsch. Vol. 9922. Lecture Notes in Computer Science. Springer, 2016, pp. 130–141. DOI: 10.1007/978-3-319-45477-1\_11. URL: [https://doi.org/10.1007/978-3-319-45477-1\\_11](https://doi.org/10.1007/978-3-319-45477-1_11).
- [121] Georg Macher et al. “SAHARA: a security-aware hazard and risk analysis method”. In: *Proceedings of the 2015 Design, Automation & Test in Europe Conference & Exhibition, DATE 2015, Grenoble, France, March 9-13, 2015*. Ed. by Wolfgang Nebel and David Atienza. ACM, 2015, pp. 621–624. URL: <http://dl.acm.org/citation.cfm?id=2755894>.
- [122] Pratyusa K. Manadhata and Jeannette M. Wing. “A Formal Model for a System’s Attack Surface”. In: *Moving Target Defense - Creating Asymmetric Uncertainty for Cyber Threats*. Ed. by Sushil Jajodia et al. Vol. 54. Advances in Information Security. Springer, 2011, pp. 1–28. DOI: 10.1007/978-1-4614-0977-9\_1. URL: [https://doi.org/10.1007/978-1-4614-0977-9\\_1](https://doi.org/10.1007/978-1-4614-0977-9_1).
- [123] Carsten Maple et al. “A Connected and Autonomous Vehicle Reference Architecture for Attack Surface Analysis”. In: *Applied Sciences* 9.23 (2019). ISSN: 2076-3417. DOI: 10.3390/app9235101. URL: <https://www.mdpi.com/2076-3417/9/23/5101>.
- [124] Kirsten Matheus and Thomas Königseder. *Automotive Ethernet*. 2nd ed. Cambridge University Press, 2017. DOI: 10.1017/9781316869543.

- 
- 
- [125] C. McCarthy, K. Harnett, and A. Carter. *Characterization of potential security threats in modern automobiles: A composite modeling approach*. National Highway Traffic Safety Administration, Oct. 2014.
- [126] Ulrike Meyer and Susanne Wetzel. “A man-in-the-middle attack on UMTS”. In: *Proceedings of the 2004 ACM Workshop on Wireless Security, Philadelphia, PA, USA, October 1, 2004*. Ed. by Markus Jakobsson and Adrian Perrig. ACM, 2004, pp. 90–97. DOI: 10.1145/1023646.1023662. URL: <https://doi.org/10.1145/1023646.1023662>.
- [127] Charlie Miller and Chris Valasek. “A survey of remote automotive attack surfaces”. In: *black hat USA 2014* (2014), p. 94.
- [128] Charlie Miller and Chris Valasek. “Remote Compromise of an Unaltered Passenger Vehicle”. In: *Black Hat USA 2015* (2015), p. 91.
- [129] *Ministry of Economics, Innovation, Digitalization and Energy of Nordrhein-Westfalen. Charge Point Market Overview*. 2022. URL: <https://www.elektromobilitaet.nrw/unsere-service/marktuebersicht-ladestationen/>.
- [130] Robert Mitchell and Ing-Ray Chen. “A survey of intrusion detection techniques for cyber-physical systems”. In: *ACM Comput. Surv.* 46.4 (2013), 55:1–55:29. DOI: 10.1145/2542049. URL: <https://doi.org/10.1145/2542049>.
- [131] Stig Fr. Mjøl̄snes and Ruxandra F. Olimid. “Easy 4G/LTE IMSI Catchers for Non-Programmers”. In: *Computer Network Security - 7th International Conference on Mathematical Methods, Models, and Architectures for Computer Network Security, MMM-ACNS 2017, Warsaw, Poland, August 28-30, 2017, Proceedings*. Ed. by Jacek Rak et al. Vol. 10446. Lecture Notes in Computer Science. Springer, 2017, pp. 235–246. DOI: 10.1007/978-3-319-65127-9\_19. URL: [https://doi.org/10.1007/978-3-319-65127-9\\_19](https://doi.org/10.1007/978-3-319-65127-9_19).
- [132] Jean-Philippe Monteouis et al. “SARA: Security Automotive Risk Analysis Method”. In: *Proceedings of the 4th ACM Workshop on Cyber-Physical System Security, CPSS@AsiaCCS 2018, Incheon, Republic of Korea, June 04-08, 2018*. Ed. by Dieter Gollmann and Jianying Zhou. ACM, 2018, pp. 3–14. DOI: 10.1145/3198458.3198465. URL: <https://doi.org/10.1145/3198458.3198465>.
- [133] Marc Muel̄tin. *RISE V2G: the Reference Implementation Supporting the Evolution of the Vehicle-2-Grid*. 2022. URL: <https://github.com/SwitchEV/RISE-V2G>.
- [134] Andreas M̄uller and Timo Lothspeich. “Plug-and-Secure Communication for CAN”. In: *CAN Newsletter* 4 (Dec. 2015), pp. 10–14.

- 
- [135] Philipp Mundhenk et al. “Lightweight authentication for secure automotive networks”. In: *Proceedings of the 2015 Design, Automation & Test in Europe Conference & Exhibition, DATE 2015, Grenoble, France, March 9-13, 2015*. Ed. by Wolfgang Nebel and David Atienza. ACM, 2015, pp. 285–288. URL: <http://dl.acm.org/citation.cfm?id=2755816>.
- [136] Myriam Neaimeh and Peter Bach Andersen. “Mind the gap- open communication protocols for vehicle grid integration”. In: *Energy Inform.* 3.1 (2020). DOI: 10.1186/s42162-020-0103-1. URL: <https://doi.org/10.1186/s42162-020-0103-1>.
- [137] Sen Nie, Ling Liu, and Yuefeng Du. “Free-fall: Hacking tesla from wireless to can bus”. In: *Briefing, Black Hat USA* (2017), pp. 1–16.
- [138] Stefan Nürnberger and Christian Rossow. “- vatiCAN - Vetted, Authenticated CAN Bus”. In: *Cryptographic Hardware and Embedded Systems - CHES 2016 - 18th International Conference, Santa Barbara, CA, USA, August 17-19, 2016, Proceedings*. Ed. by Benedikt Gierlichs and Axel Y. Poschmann. Vol. 9813. Lecture Notes in Computer Science. Springer, 2016, pp. 106–124. DOI: 10.1007/978-3-662-53140-2\\_6. URL: [https://doi.org/10.1007/978-3-662-53140-2%5C\\_6](https://doi.org/10.1007/978-3-662-53140-2%5C_6).
- [139] Hisashi Oguma et al. “New Attestation Based Security Architecture for In-Vehicle Communication”. In: *Proceedings of the Global Communications Conference, 2008. GLOBECOM 2008, New Orleans, LA, USA, 30 November - 4 December 2008*. IEEE, 2008, pp. 1909–1914. DOI: 10.1109/GLOCOM.2008.ECP.369. URL: <https://doi.org/10.1109/GLOCOM.2008.ECP.369>.
- [140] *Open Charge Point Protocol 1.6*. Version 1.6. Open Charge Alliance. 2016. URL: <http://www.openchargealliance.org/protocols/ocpp/ocpp-16/>.
- [141] *Open Charge Point Protocol 2.0.1*. Version 2.0.1. Open Charge Alliance. 2018. URL: <https://www.openchargealliance.org/protocols/ocpp-201/>.
- [142] *Open InterCharge Protocol for Emobility Service Provider*. Version 2.3. Hubject GmbH. Oct. 2020. URL: <https://github.com/hubject/oicp/releases/tag/v2.3>.
- [143] Penthertz. *V2GInjector*. 2022. URL: <https://github.com/FLUXIUS/V2GInjector>.

- 
- [144] Adrian Perrig et al. “Efficient and Secure Source Authentication for Multicast”. In: *Proceedings of the Network and Distributed System Security Symposium, NDSS 2001, San Diego, California, USA*. The Internet Society, 2001. URL: <https://www.ndss-symposium.org/ndss2001/efficient-and-secure-source-authentication-multicast/>.
- [145] Jonathan Petit and Steven E. Shladover. “Potential Cyberattacks on Automated Vehicles”. In: *IEEE Trans. Intell. Transp. Syst.* 16.2 (2015), pp. 546–556. DOI: 10.1109/TITS.2014.2342271. URL: <https://doi.org/10.1109/TITS.2014.2342271>.
- [146] Richard Petri et al. “Evaluation of lightweight TPMs for automotive software updates over the air”. In: *Embedded Security in Cars USA, escar USA 2016, Detroit, USA, June 1-2, 2016*. 2016.
- [147] Cynthia A. Phillips and Laura Painton Swiler. “A Graph-based System for Network-vulnerability Analysis”. In: *Proceedings of the 1998 Workshop on New Security Paradigms, Charlottesville, VA, USA, September 22-25, 1998*. Ed. by Bob Blakley et al. ACM, 1998, pp. 71–79. DOI: 10.1145/310889.310919. URL: <https://doi.org/10.1145/310889.310919>.
- [148] Christian Plappert and Andreas Fuchs. “Secure and Lightweight Over-the-Air Software Update Distribution for Connected Vehicles”. In: *Annual Computer Security Applications Conference, ACSAC 2023, Austin, TX, USA, December 4-8, 2023*. ACM, 2023, pp. 268–282. DOI: 10.1145/3627106.3627135. URL: <https://doi.org/10.1145/3627106.3627135>.
- [149] P Rademakers and P Klapwijk. *EV Related Protocol Study*. 2017. URL: [https://www.elaad.nl/uploads/downloads/downloads\\_download/EV\\_related\\_protocol\\_study\\_v1.1.pdf](https://www.elaad.nl/uploads/downloads/downloads_download/EV_related_protocol_study_v1.1.pdf).
- [150] Andreea-Ina Radu and Flavio D. Garcia. “LeiA: A Lightweight Authentication Protocol for CAN”. In: *Computer Security - ESORICS 2016 - 21st European Symposium on Research in Computer Security, Heraklion, Greece, September 26-30, 2016, Proceedings, Part II*. Ed. by Ioannis G. Askoxylakis et al. Vol. 9879. Lecture Notes in Computer Science. Springer, 2016, pp. 283–300. DOI: 10.1007/978-3-319-45741-3\_15. URL: [https://doi.org/10.1007/978-3-319-45741-3\\_15](https://doi.org/10.1007/978-3-319-45741-3_15).
- [151] Shahid Raza et al. “Securing communication in 6LoWPAN with compressed IPsec”. In: *Distributed Computing in Sensor Systems, 7th IEEE International Conference and Workshops, DCOSS 2011, Barcelona, Spain, 27-29 June, 2011, Proceedings*.



- 
- IEEE Computer Society, 2011, pp. 1–8. DOI: 10.1109/DCOSS.2011.5982177. URL: <https://doi.org/10.1109/DCOSS.2011.5982177>.
- [152] Eric Rescorla. *The Transport Layer Security (TLS) Protocol Version 1.3*. RFC 8446. Aug. 2018. DOI: 10.17487/RFC8446. URL: <https://www.rfc-editor.org/info/rfc8446>.
- [153] Roland Rieke. “Abstraction-based analysis of known and unknown vulnerabilities of critical information infrastructures”. In: *Int. J. Syst. Syst. Eng.* 1.1/2 (2008), pp. 59–77. DOI: 10.1504/IJSSE.2008.018131. URL: <https://doi.org/10.1504/IJSSE.2008.018131>.
- [154] Martin Ring et al. “Survey on vehicular attacks - building a vulnerability database”. In: *IEEE International Conference on Vehicular Electronics and Safety, ICVES 2015, Yokohama, Japan, November 5-7, 2015*. IEEE, 2015, pp. 208–212. DOI: 10.1109/ICVES.2015.7396919. URL: <https://doi.org/10.1109/ICVES.2015.7396919>.
- [155] Juan E. Rubio, Cristina Alcaraz, and Javier López. “Addressing Security in OCPP: Protection Against Man-in-the-Middle Attacks”. In: *9th IFIP International Conference on New Technologies, Mobility and Security, NTMS 2018, Paris, France, February 26-28, 2018*. IEEE, 2018, pp. 1–5. DOI: 10.1109/NTMS.2018.8328675. URL: <https://doi.org/10.1109/NTMS.2018.8328675>.
- [156] Alastair Ruddle et al. *Security requirements for automotive on-board networks based on dark-side scenarios*. EVITA Deliverable D2.3. last accessed 2020-01-13. EVITA project, 2009. URL: <https://evita-project.org/deliverables.html>.
- [157] David Rupperecht et al. “Breaking LTE on Layer Two”. In: *2019 IEEE Symposium on Security and Privacy, SP 2019, San Francisco, CA, USA, May 19-23, 2019*. IEEE, 2019, pp. 1121–1136. DOI: 10.1109/SP.2019.00006. URL: <https://doi.org/10.1109/SP.2019.00006>.
- [158] RWTH Aachen University. *SteVe*. 2022. URL: <https://github.com/RWTH-i5-IDSG/steve>.
- [159] SAE International. *Cybersecurity Guidebook for Cyber-Physical Vehicle Systems*. Tech. rep. J3061. SAE International, 2016.
- [160] Dietmar Scheer et al. “STAR3 - Eine neue Generation der E/E-Architektur”. In: *Sonderprojekte ATZ/MTZ* 25.1 (Dec. 2020). URL: <https://doi.org/10.1007/s41491-020-0056-5>.

- 
- [161] Christoph Schmittner et al. “Using SAE J3061 for Automotive Security Requirement Engineering”. In: *Computer Safety, Reliability, and Security - SAFECOMP 2016 Workshops, ASSURE, DECSoS, SASSUR, and TIPS, Trondheim, Norway, September 20, 2016, Proceedings*. Ed. by Amund Skavhaug et al. Vol. 9923. Lecture Notes in Computer Science. Springer, 2016, pp. 157–170. DOI: 10.1007/978-3-319-45480-1\_13. URL: [https://doi.org/10.1007/978-3-319-45480-1%5C\\_13](https://doi.org/10.1007/978-3-319-45480-1%5C_13).
- [162] Bruce Schneier. “Attack Trees: Modeling Security Threats”. In: *Dr. Dobb’s Journal* (Dec. 1999).
- [163] Sandro Schulze et al. “On the Need of Data Management in Automotive Systems”. In: *Datenbanksysteme in Business, Technologie und Web (BTW 2009), 13. Fachtagung des GI-Fachbereichs “Datenbanken und Informationssysteme” (DBIS), Proceedings, 2.-6. März 2009, Münster, Germany*. Ed. by Johann Christoph Freytag et al. Vol. P-144. LNI. GI, 2009, pp. 217–226. URL: <https://dl.gi.de/20.500.12116/20446>.
- [164] Hendrik Schweppe et al. *Deliverable D3.3: Secure On-Board Protocols Specification*. Tech. rep. EVITA, July 2011.
- [165] M. Seaman. *MACsec in the car*. <http://www.ieee802.org/1/files/public/docs2016/ae-seaman-macsec-in-the-car-0716-v0.pdf>. 2017.
- [166] Altaf Shaik et al. “New vulnerabilities in 4G and 5G cellular access network protocols: exposing device capabilities”. In: *Proceedings of the 12th Conference on Security and Privacy in Wireless and Mobile Networks, WiSec 2019, Miami, Florida, USA, May 15-17, 2019*. ACM, 2019, pp. 221–231. DOI: 10.1145/3317549.3319728. URL: <https://doi.org/10.1145/3317549.3319728>.
- [167] Ali Shuja Siddiqui et al. “Secure communication over CANBus”. In: *IEEE 60th International Midwest Symposium on Circuits and Systems, MWSCAS 2017, Boston, MA, USA, August 6-9, 2017*. IEEE, 2017, pp. 1264–1267. DOI: 10.1109/MWSCAS.2017.8053160. URL: <https://doi.org/10.1109/MWSCAS.2017.8053160>.
- [168] Nadine Sinner et al. *Anforderungsanalyse für Selbstschutz im vernetzten Fahrzeug*. 2017. URL: <https://publica.fraunhofer.de/handle/publica/298839>.
- [169] Chawin Sitawarin et al. “Rogue Signs: Deceiving Traffic Sign Recognition with Malicious Ads and Logos”. In: *CoRR abs/1801.02780 (2018)*. arXiv: 1801.02780. URL: <http://arxiv.org/abs/1801.02780>.

- 
- [170] Saleh Soltan, Prateek Mittal, and H. Vincent Poor. “BlackIoT: IoT Botnet of High Wattage Devices Can Disrupt the Power Grid”. In: *27th USENIX Security Symposium, USENIX Security 2018, Baltimore, MD, USA, August 15-17, 2018*. Ed. by William Enck and Adrienne Porter Felt. USENIX Association, 2018, pp. 15–32. URL: <https://www.usenix.org/conference/usenixsecurity18/presentation/soltan>.
- [171] Florian Sommer, Jürgen Dürrwang, and Reiner Kriesten. “Survey and Classification of Automotive Security Attacks”. In: *Information* 10.4 (2019). ISSN: 2078-2489. DOI: 10.3390/info10040148. URL: <https://www.mdpi.com/2078-2489/10/4/148>.
- [172] Florian Sommer et al. “Combining Cyber Security Intelligence to Refine Automotive Cyber Threats”. In: *ACM Trans. Priv. Secur.* (Feb. 2024). Just Accepted. ISSN: 2471-2566. DOI: 10.1145/3644075. URL: <https://doi.org/10.1145/3644075>.
- [173] Dieter Spaar. “Beemer, Open Thyself! - Security vulnerabilities in BMW’s ConnectedDrive”. In: *c’t 05/2015* (2015). <http://heise.de/-2540957>.
- [174] Vangelis Stykas. *Smart car chargers. Plug-n-Play for hackers?* July 2021. URL: <https://www.pentestpartners.com/security-blog/smart-car-chargers-plug-n-play-for-hackers/>.
- [175] A. Sundaresan, L. Gerard, and M. Kim. *Secure ROS*. <http://secure-ros.csl.sri.com/>. 2017.
- [176] *TPM Main Specification Level 2 Version 1.2*. Standard. Beaverton, Oregon: Trusted Computing Group, Mar. 2011. URL: <https://trustedcomputinggroup.org/resource/tpm-main-specification/>.
- [177] The EVITA consortium. *EVITA Threat and Risk Analysis*. <https://www.evita-project.org>. last accessed 2020-01-13. Dec. 2009.
- [178] The Tamarin Team. *Tamarin-Prover Manual - Security Protocol Analysis in the Symbolic Model*. 2019.
- [179] M. Thomson. *Cipher Suites for Negotiating Zero Round Trip (0-RTT) Transport Layer Security (TLS) with Renewed Certificate Authentication*. Internet-Draft draft-thomson-tls-0rtt-and-certs-01. Work in Progress. Internet Engineering Task Force, May 2016. 7 pp. URL: <https://tools.ietf.org/html/draft-thomson-tls-0rtt-and-certs-01>.

- 
- [180] Bernd Thormann et al. *Evaluation of Grid Relieving Measures for Integrating Electric Vehicles in a Suburban Low-Voltage Grid*. Madrid, Spain, June 2019. DOI: <http://dx.doi.org/10.34890/398>. URL: <https://cired-repository.org/handle/20.500.12455/205>.
- [181] Hannes Tschofenig and Pasi Eronen. *Pre-Shared Key Ciphersuites for Transport Layer Security (TLS)*. RFC 4279. Dec. 2005. DOI: [10.17487/RFC4279](https://doi.org/10.17487/RFC4279). URL: <https://www.rfc-editor.org/info/rfc4279>.
- [182] UAG „Standard Data Protection Model“ of the AK Technik of the Independent Data Protection Supervisory Authorities of the Federation and the Länder, ed. *The Standard Data Protection Model - A method for Data Protection advising and controlling on the basis of uniform protection goals*. Version 3.0a. Nov. 2022. URL: [https://www.datenschutz-mv.de/static/DS/Dateien/Datenschutzmodell/SDM\\_V3\\_en.pdf](https://www.datenschutz-mv.de/static/DS/Dateien/Datenschutzmodell/SDM_V3_en.pdf).
- [183] Hiroshi Ueda et al. “Security authentication system for in-vehicle network”. In: *SUMITOMO ELECTRIC TECHNICAL REVIEW* 81 (Oct. 2015), pp. 5–9.
- [184] UN Regulation No. 155. *Uniform provisions concerning the approval of vehicles with regards to cyber security and cyber security management system*. [Online; accessed 30-Apr-2021]. United Nations, Mar. 2021. URL: <https://unece.org/sites/default/files/2021-03/R155e.pdf>.
- [185] UN Regulation No. 156. *Uniform provisions concerning the approval of vehicles with regards to software update and software updates management system system*. [Online; accessed 30-Apr-2021]. United Nations, Mar. 2021. URL: <https://unece.org/sites/default/files/2021-03/R156e.pdf>.
- [186] Andreas Unterweger et al. “Low-risk Privacy-preserving Electric Vehicle Charging with Payments”. In: *Workshop on Automotive and Autonomous Vehicle Security (AutoSec) 2021 25 February 2021, Virtual*. Feb. 2021. DOI: [10.14722/autosec.2021.23001](https://doi.org/10.14722/autosec.2021.23001).
- [187] Anthony Van Herrewege, Dave Singelée, and Ingrid Verbauwhede. “CANAuth - A Simple, Backward Compatible Broadcast Authentication Protocol for CAN bus”. In: *9th Embedded Security in Cars (escar) Conference Europe 2011*. Vol. 9. International School of IT Security AG. Dresden, Germany, Nov. 2011.
- [188] Vector. *CCS Listener*. 2022. URL: <https://www.vector.com/int/en/products/products-a-z/hardware/vh5110/>.

- 
- [189] Marco De Vincenzi et al. “A Systematic Review on Security Attacks and Countermeasures in Automotive Ethernet”. In: *ACM Comput. Surv.* 56.6 (2024), 135:1–135:38. DOI: 10.1145/3637059. URL: <https://doi.org/10.1145/3637059>.
- [190] W. Voss. *A Comprehensible Guide to Controller Area Network*. Copperhill Technologies Corporation, 2008. ISBN: 9780976511601. URL: <https://books.google.de/books?id=PU6pp03XbUwC>.
- [191] Jie Wang et al. “Unified Parametrizable Attack Tree”. In: *International Journal for Information Security Research (IJISR)* 1 (1/2 2011), pp. 20–26.
- [192] Qiyan Wang and Sanjay Sawhney. “VeCure: A practical security framework to protect the CAN bus of vehicles”. In: *4th International Conference on the Internet of Things, IOT 2014, Cambridge, MA, USA, October 6-8, 2014*. IEEE, 2014, pp. 13–18. DOI: 10.1109/IOT.2014.7030108. URL: <https://doi.org/10.1109/IOT.2014.7030108>.
- [193] Stephan Wesemeyer et al. “Formal Analysis and Implementation of a TPM 2.0-based Direct Anonymous Attestation Scheme”. In: *ASIA CCS '20: The 15th ACM Asia Conference on Computer and Communications Security, Taipei, Taiwan, October 5-9, 2020*. Ed. by Hung-Min Sun et al. ACM, 2020, pp. 784–798. DOI: 10.1145/3320269.3372197. URL: <https://doi.org/10.1145/3320269.3372197>.
- [194] Jordan Whitefield et al. “Formal Analysis of V2X Revocation Protocols”. In: *Security and Trust Management - 13th International Workshop, STM 2017, Oslo, Norway, September 14-15, 2017, Proceedings*. Ed. by Giovanni Livraga and Chris J. Mitchell. Vol. 10547. Lecture Notes in Computer Science. Springer, 2017, pp. 147–163. DOI: 10.1007/978-3-319-68063-7\_10. URL: [https://doi.org/10.1007/978-3-319-68063-7\\_10](https://doi.org/10.1007/978-3-319-68063-7_10).
- [195] Marcel Wille and Olaf Krieger. *Ethernet & Adaptive AUTOSAR Key elements of the new Volkswagen E/E architecture*. May 2017.
- [196] Marian Willuhn. *Satellite cyber attack paralyzes 11GW of German wind turbines*. Mar. 2022. URL: <https://www.pv-magazine.com/2022/03/01/satellite-cyber-attack-paralyzes-11gw-of-german-wind-turbines/>.
- [197] Marko Wolf, André Weimerskirch, and Christof Paar. “Security in Automotive Bus Systems”. In: *2nd Embedded Security in Cars (escar) Conference Europe 2004*. Vol. 2. International School of IT Security AG. Bochum, Germany, Nov. 2004.

- 
- [198] Samuel Woo et al. “A Practical Security Architecture for In-Vehicle CAN-FD”. In: *IEEE Trans. Intell. Transp. Syst.* 17.8 (2016), pp. 2248–2261. DOI: 10.1109/TITS.2016.2519464. URL: <https://doi.org/10.1109/TITS.2016.2519464>.
- [199] Wenqian Xin et al. “Improved Cryptanalysis on SipHash”. In: *Cryptology and Network Security - 18th International Conference, CANS 2019, Fuzhou, China, October 25-27, 2019, Proceedings*. Ed. by Yi Mu, Robert H. Deng, and Xinyi Huang. Vol. 11829. Lecture Notes in Computer Science. Springer, 2019, pp. 61–79. DOI: 10.1007/978-3-030-31578-8\_4. URL: [https://doi.org/10.1007/978-3-030-31578-8\\_4](https://doi.org/10.1007/978-3-030-31578-8_4).
- [200] Minrui Yan, Jiahao Li, and Guy Harpak. “Security Research on Mercedes-Benz: From Hardware to Car Control”. In: *Black Hat USA* (2020).
- [201] Ye Yan et al. “A Survey on Cyber Security for Smart Grid Communications”. In: *IEEE Commun. Surv. Tutorials* 14.4 (2012), pp. 998–1010. DOI: 10.1109/SURV.2012.010912.00035. URL: <https://doi.org/10.1109/SURV.2012.010912.00035>.
- [202] Daniel Zelle, Sigrid Gürgens, and Anjia Yang. “BusCount: A Provable Replay Protection Solution for Automotive CAN Networks”. In: *Security and Communication Networks 2021* (Jan. 2021). ISSN: 1939-0114. DOI: 10.1155/2021/9951777. URL: <https://doi.org/10.1155/2021/9951777>.
- [203] Daniel Zelle et al. “Analyzing and Securing SOME/IP Automotive Services with Formal and Practical Methods”. In: *ARES 2021: The 16th International Conference on Availability, Reliability and Security, Vienna, Austria, August 17-20, 2021*. Ed. by Delphine Reinhardt and Tilo Müller. ACM, 2021, 8:1–8:20. DOI: 10.1145/3465481.3465748. URL: <https://doi.org/10.1145/3465481.3465748>.
- [204] Daniel Zelle et al. “Anonymous Charging and Billing of Electric Vehicles”. In: *Proceedings of the 13th International Conference on Availability, Reliability and Security, ARES 2018, Hamburg, Germany, August 27-30, 2018*. Ed. by Sebastian Doerr et al. ACM, 2018, 22:1–22:10. ISBN: 978-1-4503-6448-5. DOI: 10.1145/3230833.3230850. URL: <https://doi.org/10.1145/3230833.3230850>.
- [205] Daniel Zelle et al. “GEGENÜBERSTELLUNG VON LADEEINRICHTUNGSARCHITEKTURENTWÜRFEN FÜR EICHRECHTSKONFORME AUTHENTIFIZIERTE LADEVORGÄNGE UNTER EINBEZIEHUNG EINES SMGWS”. In: (Dec. 2021). URL: [https://www.sit.fraunhofer.de/fileadmin/dokumente/studien\\_und\\_technical\\_reports/Whitepaper\\_LamA.pdf?\\_=1655801427](https://www.sit.fraunhofer.de/fileadmin/dokumente/studien_und_technical_reports/Whitepaper_LamA.pdf?_=1655801427).

- 
- [206] Daniel Zelle et al. "On Using TLS to Secure In-Vehicle Networks". In: *Proceedings of the 12th International Conference on Availability, Reliability and Security, Reggio Calabria, Italy, August 29 - September 01, 2017*. ACM, 2017, 67:1–67:10. ISBN: 978-1-4503-5257-4. DOI: 10.1145/3098954.3105824. URL: <https://doi.org/10.1145/3098954.3105824>.
- [207] Daniel Zelle et al. "SEPAD - Security Evaluation Platform for Autonomous Driving". In: *28th Euromicro International Conference on Parallel, Distributed and Network-Based Processing, PDP 2020, Västerås, Sweden, March 11-13, 2020*. IEEE, 2020, pp. 413–420. DOI: 10.1109/PDP50117.2020.00070. URL: <https://doi.org/10.1109/PDP50117.2020.00070>.
- [208] Daniel Zelle et al. "ThreatSurf: A method for automated Threat Surface assessment in automotive cybersecurity engineering". In: *Microprocessors and Microsystems* 90 (2022), p. 104461. ISSN: 0141-9331. DOI: <https://doi.org/10.1016/j.micpro.2022.104461>. URL: <https://www.sciencedirect.com/science/article/pii/S0141933122000321>.
- [209] Kim Zetter. *Inside the Cunning, Unprecedented Hack of Ukraine's Power Grid*. Mar. 2016. URL: <https://www.pv-magazine.com/2022/03/01/satellite-cyber-attack-paralyzes-11gw-of-german-wind-turbines/>.
- [210] Tianyu Zhao et al. "A secure and privacy-preserving payment system for Electric vehicles". In: *2015 IEEE International Conference on Communications, ICC 2015, London, United Kingdom, June 8-12, 2015*. IEEE, 2015, pp. 7280–7285. DOI: 10.1109/ICC.2015.7249489. URL: <https://doi.org/10.1109/ICC.2015.7249489>.
- [211] Maria Zhdanova. "Security and Trust in Safety Critical Infrastructures". PhD thesis. Technical University of Darmstadt, Germany, 2022. URL: <http://tuprints.ulb.tu-darmstadt.de/21505/>.
- [212] Maria Zhdanova et al. "Local Power Grids at Risk - An Experimental and Simulation-based Analysis of Attacks on Vehicle-To-Grid Communication". In: *Annual Computer Security Applications Conference, ACSAC 2022, Austin, TX, USA, December 5-9, 2022*. ACM, 2022, pp. 42–55. DOI: 10.1145/3564625.3568136. URL: <https://doi.org/10.1145/3564625.3568136>.
- [213] Tobias Ziermann, Stefan Wildermann, and Jürgen Teich. "CAN+: A new backward-compatible Controller Area Network (CAN) protocol with up to 16× higher data rates". In: *Design, Automation and Test in Europe, DATE 2009, Nice, France, April 20-24, 2009*. Ed. by Luca Benini et al. IEEE, 2009, pp. 1088–1093. DOI: 10.110

---

9/DATE.2009.5090826. URL: <https://doi.org/10.1109/DATE.2009.5090826>.

- [214] Qingwu Zou et al. "The Study of Secure CAN Communication for Automotive Applications". In: *WCX<sup>TM</sup> 17: SAE World Congress Experience*. SAE International, Mar. 2017. DOI: <https://doi.org/10.4271/2017-01-1658>. URL: <https://doi.org/10.4271/2017-01-1658>.





---

## A. Appendix

---

---

## A.1. Formal Security of CAN

We suggested using the well-established Tamarin prover [12, 178] in the work [110] for automotive protocols.

### A.1.1. Formal Security Analysis of Generic Counter Concept

On the basis of the generic security protocol for CAN presented in Section 5.2, we designed a formal model with the well-established Tamarin prover [12, 178].

A Tamarin model consists of states and state transitions, which are used to check security properties defined as lemmas.

First, we first defined the state transitions for the generic counter model and after that, we transferred the security properties into the Tamarin model. Finally, the security properties are solved by the Tamarin prover. This will identify possible weaknesses or prove the security of the protocol.

#### State Transitions

The state transitions are also called rules in Tamarin. We start with the initialization of a random secret key which will stay for this model (cf. Listing A.1 line 1). The key can get compromised with the rule *CompromiseKey* (cf. Listing A.1 line 4). Furthermore, the sender (cf. Listing A.1 line 7) and receiver ECU (cf. Listing A.1 line 11) are initialized with a name, a local counter, a phase (that emulates the time), and the current status.

Listing A.1: Initialisation of Generic Counter

```
1     rule RegisterKey:
2     [ Fr(~k) ] --[ KeyGen(~k), OnlyOnce('KeyGen') ]-> [ !Key(~k) ]
3
4     rule CompromiseKey:
5     [ !Key(~k) ] --[ Compromise() ]-> [ Out(~k) ]
6
7     rule InitSender:
8     [ !Key(~k) ] --[ Sender($A), OnlyOnce(<'Sender'>) ]->
9     [ ECU($A, '1', '1', 'Send') ]
10
11    rule InitReceiver:
```

---

```

12 [ !Key(~k) ] --[ Receiver($B), OnlyOnce(<'Receiver'>) ]->
13 [ ECU($B, '1', '1', 'Empty') ]

```

Next, a message needs to be sent from an ECU (cf. Listing A.2) using its local counter  $c_a$ , the message content  $m$  and the key  $k$  to calculate a MAC (line 5). Furthermore, every CAN message has a CRC checksum over the message content.

#### Listing A.2: Send CAN Message

```

1 rule SendMessage:
2 let m = 'messege' in
3 [ ECU($A, c_a, phase, 'Send'), !Key(~k) ]
4 --[ Send($A, $m, phase), OnlyOnce('Send') ]->
5 [ Out(<'Send', $A, $m, c_a+'1', h(<$m, c_a+'1', ~k>), h($m)>) ]

```

Another ECU receives (cf. Listing A.3) this message and checks for the integrity of the CRC checksum. If the message is correct from an integrity point of view it will get processed (lines 1-4) otherwise it is dropped (lines 6-9). Note: Once a receiving process ends by an error or by accepting the message, the phase is increased by one to emulate the end of a time period. This allows us to track whether a message is received in the same time period it was sent.

#### Listing A.3: Receive CAN Message

```

1 rule ReceiveMessage:
2 [ ECU($B, c_b, phase, 'Empty'), In(<'Send', $A, $m, c_m, mac, crc>),
3 !Key(~k) ]
4 --[ Receive($B, $m, phase), Neq($A, $B), Eq(crc, h($m)) ]->
5 [ ECU($B, c_b, phase, <'Send', $A, $m, c_m, mac, crc>) ]
6 rule ReceiveMessageError:
7 [ ECU($B, c_b, phase, 'Empty'), In(<'Send', $A, $m, c_m, mac, crc>) ]
8 --[ ReceiveE($B, $m, phase), Neq($A, $B), Neq(crc, h($m)) ]->
9 [ ECU($B, c_b, phase+'1', 'Empty') ]

```

In case the checksum of a message is correct the ECU evaluates the correctness of the MAC and the validity of the transmitted counter value as described in the previous section. This check can trigger three possible rules (cf. Listing A.4). First the check is correct and the message is processed (lines 1-4), second the check fails due to not matching mac values (lines 6-9), and third the check fails due to an incorrect counter value (lines 11-14).

---

#### Listing A.4: Processing CAN Message

```
1 rule ProcessMessage :
2 [ ECU($B,c_b, phase, <'Send', $A, $m, c_m, mac, crc>), !Key(~k)]
3 --[ Process($B,$m,phase), LessThan(c_b,c_m), Eq(mac,h($m,c_m,~k))
4 ]->
5 [ ECU($B,c_m,phase+'1', 'Empty')]
6
7 rule ProcessError1 :
8 [ ECU($B,c_b, phase, <'Send', $A, $m, c_m, mac, crc>), !Key(~k)]
9 --[ ProcessE($B,$m,phase), Neq($A,$B), Neq(mac,h($m,c_m,~k)) ]->
10 [ ECU($B,c_b, phase+'1', 'Empty')]
11
12 rule ProcessError2 :
13 [ ECU($B,c_b, phase, <'Send', $A, $m, c_m, mac, crc>), !Key(~k)]
14 --[ ProcessE($B,$m,phase), Neq($A,$B), GreaterEq(c_b,c_m) ]->
15 [ ECU($B,c_b, phase+'1', 'Empty')]
```

We have modeled the security goals defined in Section 5.1.1 into the Tamarin model.

**Data Authenticity** requires the message to be unaltered during the transmission of the message. For every message that is received an identical message needs to be sent via the CAN bus.

#### Listing A.5: Data Authenticity

```
1 lemma DataAuthenticity :
2 "All receiver msg pha2 #t1 #t3 .
3 Receiver(receiver) @ t1
4 & Process(receiver,msg,pha2) @ t3
5 ==> ((Ex sender pha1 #t0 #t2 . Sender(sender) @ t0
6 & Send(sender,msg,pha1) @ t2)
7 | (Ex #r . Compromise() @r))"
```

**Non-repeatability** a message which is accepted only once. So an attacker cannot replay a valid message.

#### Listing A.6: Non-Repetition

```
1 lemma NonRepetition :
2 "All receiver msg1 msg2 pha1 pha2 #t1 #t3 #t2 .
```

---

```

3 Receiver(receiver) @ t1
4 & Process(receiver ,msg1,pha1) @ t2
5 & Process(receiver ,msg2,pha2) @ t3
6 ==> ((msg1=msg2)
7 | (Ex #r . Compromise() @r))”

```

**Immediacy** The property of immediacy is given if a message is sent, received, and accepted in the same time phase.

#### Listing A.7: Immediacy

```

1 lemma Immediacy:
2 ”All sender receiver msg pha1 pha2 #t0 #t1 #t3 #t2 .
3 Sender(sender) @ t0
4 & Receiver(receiver) @ t1
5 & Send(sender ,msg,pha1) @ t2
6 & Process(receiver ,msg,pha2) @ t3
7 ==> ((pha1=pha2)
8 | (Ex #r . Compromise() @r))”

```

## Results of the Formal Analysis

Within the formal model, the generic approach fulfills the properties of data origin authenticity and non-repeatability. Both are proven within the Tamarin model. But the prover could also identify an attack that violates the property of immediacy. An attacker can invalidate a message and repeat it in a different time phase. As a result, the receiver cannot be certain that the received message was sent within a defined time frame and may be delayed by the attacker.

### A.1.2. Formal Verification of BusCount

Similar to the generic counter model, we also verified the BusCount protocol introduced in Section 5.3 using Tamarin. For the purpose of this analysis, we divide the protocol into two parts, the actual message transfer, and the synchronization.

---

## Message Transfer BusCount

In the following we describe the model of the BusCount message protocol in three steps, initialization (Listing A.8), sending a message (Listing A.9), and receiving a message (Listing A.10).

For the initialization, a random key  $k$  is generated and the sender and receiver are initialized with the same counter value  $i$ . Note: We discuss the situation of different counter values in the synchronization part. Furthermore, we introduce a state transition allowing the attacker to obtain the key  $k$ .

Listing A.8: Initialisation of BusCount

```
1   rule RegisterKey:
2   [ Fr(~k) ]
3   --[ KeyGen(~k), OnlyOnce('KeyGen') ]->
4   [ !Key(~k) ]
5
6   rule InitCounter:
7   [] --> [ Counter('1') ]
8
9   rule IncCounter:
10  [ Counter(counter) ]
11  -->
12  [ Counter(counter+'1') ]
13
14  rule InitECUa:
15  [ !Key(~k), Counter(ca)]
16  --[ ECU1($A, ca), OnlyTwice(<'ECU'>) ]->
17  [ !ECU($A, ca) ]
18
19  rule CompromiseKey:
20  [ !Key(~k) ]
21  --[ Compromise() ]->
22  [ Out(~k) ]
```

The send transaction increments the local counter of the sender and calculates a keyed hash over the message and the counter. Note: In contrast to the described protocol the Tamarin model increases counters due to the limitations of Tamarin. Nevertheless, it is identical to decrementing the counter. Furthermore, a hash over the message represents the CRC of the CAN message. A receiver receives a message where we make sure that the receiver and sender are not identical.

---

### Listing A.9: Send and Recieve BusCount Message

```
1 rule SendMessage:
2   let m = 'messege' in
3   [ ECU($A,c_a, phase, 'Send'), !Key(~k)]
4   --[ Send($A,$m,phase), OnlyOnce('Send') ]->
5   [ Out(<'Send', $A, $m, c_a+'1', h(<$m, c_a+'1', ~k>), h($m)>) ]
6
7
8 rule ReceiveMessage:
9   [ ECU($B,c_b, phase, 'Empty'), In(<'Send', $A, $m, c_m, mac, crc>),
10    !Key(~k)]
11  --[ Receive($B,$m,phase), Neq($A,$B) ]->
12  [ ECU($B,c_b+'1', phase, <'Send', $A, $m, c_m, mac, crc>)]
```

In the following step the receiver makes sure that MAC and CRC are both correct. Only if this is the case the receiver processes the message. In any case the phase is incremented by one. This represents time passing which is used for the verification to determine the time.

### Listing A.10: Process BusCount Message

```
1 rule ProcessMessage:
2   [ ECU($B,c_b, phase, <'Send', $A, $m, c_m, mac, crc>), !Key(~k)]
3   --[ Process($B,$m,phase), Eq(mac,h(<$m,c_b,~k>)), Eq(crc,h($m)) ]
4   ->
5   [ ECU($B,c_b, phase+'1', 'Empty')]
6
7 rule ProcessError0:
8   [ ECU($B,c_b, phase, <'Send', $A, $m, c_m, mac, crc>)]
9   --[ ProcessE($B,$m,phase), Neq($A,$B), Neq(crc,h($m)) ]->
10  [ ECU($B,c_b, phase+'1', 'Empty')]
11
12 rule ProcessError1:
13  [ ECU($B,c_b, phase, <'Send', $A, $m, c_m, mac, crc>), !Key(~k)]
14  --[ ProcessE($B,$m,phase), Neq($A,$B), Neq(mac,h(<$m,c_b,~k>)) ]->
15  [ ECU($B,c_b, phase+'1', 'Empty')]
```

**Data Authenticity** requires the message to be unaltered during the transmission of the message. For every message that is received an identical message needs to be sent via the

---

CAN bus.

Listing A.11: Data Authenticity

```
1 lemma DataAuthenticity :
2   "All receiver msg pha2 #t1 #t3 .
3   Receiver(receiver) @ t1
4   & Process(receiver ,msg,pha2) @ t3
5   ==> ((Ex sender pha1 #t0 #t2 . Sender(sender) @ t0
6   & Send(sender ,msg,pha1) @ t2)
7   | (Ex #r . Compromise() @r))"
```

**Non-repeatability** a message which is accepted only once. So an attacker cannot replay a valid message.

Listing A.12: Non-Repetition

```
1 lemma NonRepetition :
2   "All receiver msg1 msg2 pha1 pha2 #t1 #t3 #t2 .
3   Receiver(receiver) @ t1
4   & Process(receiver ,msg1,pha1) @ t2
5   & Process(receiver ,msg2,pha2) @ t3
6   ==> ((pha1=pha2)
7   | (Ex #r . Compromise() @r))"
```

**Immediacy** The property of immediacy is given if a message is sent, received, and accepted in the same time phase.

Listing A.13: Immediacy

```
1 lemma Immediacy :
2   "All sender receiver msg pha1 pha2 #t0 #t1 #t3 #t2 .
3   Sender(sender) @ t0
4   & Receiver(receiver) @ t1
5   & Send(sender ,msg,pha1) @ t2
6   & Process(receiver ,msg,pha2) @ t3
7   ==> ((pha1=pha2)
8   | (Ex #r . Compromise() @r))"
```



---

## BusCount Synchronization

For the synchronization mechanism, two ECUs are initialized with an arbitrary counter. Both start the synchronization process by sending out their local counter. Dependent on the local counter values of both ECUs the ECU with the largest counter sends a keyed hash value over its local counter. A receiver validates this message and updates it to the transmitted counter.

Listing A.14: Synchronisation

```
1 rule SendSyncCounter:
2 [ !ECU($A, c_a), !ECU($B, c_b) ]
3 --[ SyncSend($A, (c_a+'1'), $B, (c_b+'1')), OnlyOnce(<$A, 'Sync'>),
4 Neq($A, $B) ]->
5 [ Out(<'SyncMsg', $A, c_a+'1', $B, c_b+'1'>) ]
6
7 rule SendSyncAuthA:
8 [ !ECU($A, c_a), !ECU($B, c_b), !Key(~k), In(<'SyncMsg', $A, c_a1,
9 $B, c_b1>) ]
10 --[ SyncAuth($A, c_a1), GreaterEq(c_a1, c_b1), Neq($A, $B), Eq(
11 c_a+'1', c_a1), Eq(c_b+'1', c_b1) ]->
12 [ Out(<'SyncMsgAuth', c_a1, h(<c_a1, ~k>)>) ]
13
14 rule SendSyncAuthB:
15 [ !ECU($A, c_a), !ECU($B, c_b), !Key(~k), In(<'SyncMsg', $A, c_a1,
16 $B, c_b1>) ]
17 --[ SyncAuth($B, c_b1), LessThan(c_a1, c_b1), Neq($A, $B), Eq(c_a
18 +'1', c_a1), Eq(c_b+'1', c_b1) ]->
19 [ Out(<'SyncMsgAuth', c_b1, h(<c_b1, ~k>)>) ]
20
21 rule ProcessSync:
22 [ !Key(~k), In(<'SyncMsgAuth', c_a, mac_a>) ]
23 --[ Eq(mac_a, h(c_a, ~k))] ->
24 [ SyncTo(c_a) ]
25
26 rule ProcessSyncDone:
27 [ SyncTo(c_a) ] - [ SyncDone(c_a) ] -> [ ]
```

For the verification we evaluated whether an attacker is capable of performing a sync without the knowledge of the key.

Listing A.15: Correct Synchronisation

---

```

1 lemma SecureSync:
2   "All c #t0.
3   SyncDone(c) @ t0
4   => ((Ex ecu1 ecu2 c1 c2 #t1 #t2.
5   ECU1(ecu1,c1) @ t1 & ECU1(ecu2,c2) @ t2 & ((c1+'1'=c) & ((c1=c2 |
6     Ex z. c1+z=c2)) | ((c2+'1'=c) & (Ex z. c2+z=c1))))
7   | (Ex #r. Compromise() @r))"
8 end

```

Note: An attacker is still able to interrupt the transmission of a synchronization.

### Results of the Formal Analysis

Based on the formal model all security properties can be proven to be correct. However, it is still possible for attackers to perform denial of service attacks.

---

## A.2. Formal Security of SOME/IP

In the following, we provide the Tamarin models we used to analyze SOME/IP as well as our proposed security extensions SESO-RC and SESO-AS.

### A.2.1. Tamarin model of SOME/IP

As described in Section 7.4, our Tamarin model of SOME/IP focuses on the authentication properties of service discovery messages. Thus, to reduce the model's complexity, it does not implement *Publish/Subscribe* communication. However, the identified weaknesses apply to all communication patterns and, thus, the attack can be extended for *Publish/Subscribe* communication as shown in Section 6.3.4. In addition, the session ID counters have been abstracted by nonces.

The underlying security protocol is abstracted by a *group-secure* channel. Only legitimate entities can send or receive messages on this channel and messages cannot be replayed. This corresponds to an authenticated and encrypted communication. Note, that in many cases, only authentication will be used. Thus, the adversary could read but not manipulate the communication. However, we only consider the most restrictive case as we are looking for practical attacks instead of security verification. Similarly, the channel models point-to-point and broadcast communication instead of a bus architecture, as this is more suitable for Automotive Ethernet.

The first set of security lemmas considers the case that the adversary has access to the network but has not corrupted any legitimate ECU and thus, is prevented from manipulation of the network. Next, we consider the case where the adversary is allowed to corrupt ECUs. The model allows for corruption of a single ECU only to get more practical and simple attacks. The last two lemmas correspond to the copycat and de-association attacks described in Section 6.3.4 and 6.3.4.

Listing A.16: My Caption

```
1 theory Someip
2 begin
3 /*
4 Public Variables
5     $ServiceID – identifier of a service
6     $InstanceID – identifier of a instance offering a service
7     $Endpoint – IP + TCP/UDP address information
```

---

```

8         (S_/C_ prefix for server/client)
9     $MethodID – method identifier within a service
10    $ClientID – client identifier within ECU
11
12 Variables
13     SessionID – enables to differentiate between previous requests,
14                 abstracted by a nonce
15
16 Facts:
17     !S_Identity(<$ServiceID, $InstanceID, $S_Endpoint>)
18         – server identity information
19     S_State(<$ServiceID, $InstanceID, $Endpoint>,
20             'open')
21         – server is currently offering its service
22     S_OpenSubscription(<$ServiceID, $InstanceID, $S_Endpoint>, <$
23                         ClientID, SessionID>)
24         – Open Subscription Request from Client
25     S_SubscribedClient(<$ServiceID, $InstanceID, $S_Endpoint>, <$
26                         ClientID, SessionID, $C_Endpoint>, $EventgroupID)
27         – Connection data of a client subscribed to the server
28     !C_Identity(<$ClientID, $C_Endpoint>)
29         – Client identity information
30     C_Available(<$ClientID, $C_Endpoint>,
31                <$ServiceID, $InstanceID, $S_Endpoint>)
32         – Client has previously received a service offer for this
33           service
34     C_OpenSubscription(<$ClientID, $C_Endpoint>,
35                        <$ServiceID, $InstanceID, $S_Endpoint>,
36                        $EventgroupID, SessionID)
37         – Client has an open subscription request at server
38     C_Subscription(<$ClientID, $C_Endpoint>,
39                   <$ServiceID, $InstanceID, $S_Endpoint>,
40                   $EventgroupID, SessionID)
41         – Client has an active subscription at server
42 */
43 // **** Service Discovery ****
44 // ** Server **
45 rule CreateServer:
46     let
47         server = <$ServiceID, $InstanceID, $S_Endpoint>
48     in

```

---

```

46 []
47 --[ OnlyOnce(<'create_server', $ServiceID,
48           $InstanceID>),
49       OnlyOnce(<'create_server', $S_Endpoint>),
50       E_S_Created(server)
51 ]->
52 [ !S_Identity(server) ]
53
54 rule SendOfferService:
55 let
56     server = <$ServiceID, $InstanceID, $S_Endpoint>
57 in
58 [ !S_Identity(server), Fr(~nonce) ]
59 --[ E_S_SendOfferService(server, ~nonce),
60     E_Send(<'offer_service', server, ~nonce>)
61 ]->
62 [ Out_P('broadcast', <'offer_service', server,
63         ~nonce>),
64     S_State(server, 'open')
65 ]
66
67 rule SendStopService:
68 let
69     server = <$ServiceID, $InstanceID, $S_Endpoint>
70 in
71 [ !S_Identity(server),
72   S_State(server, 'open'),
73   Fr(~nonce)
74 ]
75 --[ E_S_SendStopOfferService(server, ~nonce),
76     E_S_OpenServiceOffer(server),
77     E_Send(<'stop_service', server, ~nonce>)
78 ]->
79 [ Out_P('broadcast', <'stop_service', server,
80         ~nonce>) ]
81
82 rule ReceiveAndAnswerFindService:
83 let
84     server = <$ServiceID, $InstanceID, $S_Endpoint>
85 in
86 [ !S_Identity(server),
87   S_State(server, 'open'),

```

---

```

88     In_P($S_Endpoint, <'find_service', $ServiceID>, nonce)
89 ]
90 --[ OnlyOnce(<server, nonce>),
91     E_S_OpenServiceOffer(server),
92     E_Send(<'offer_service', server>)
93 ]->
94 [   Out_P('broadcast', <'offer_service', server>), S_State(server, '
      open') ]
95
96 // ** Client **
97 rule CreateClient:
98 []
99 --[ OnlyOnce(<'create_client', $ClientID>),
100     OnlyOnce(<'create_client', $C_Endpoint>)
101 ]->
102 [   !C_Identity(<$ClientID, $C_Endpoint>) ]
103
104 rule SendFindService:
105 let
106     client = <$ClientID, $C_Endpoint>
107 in
108 [   !C_Identity(client) ]
109 --[ E_Send(<'find_service', $ServiceID>) ]->
110 [   Out_P('broadcast', <'find_service', $ServiceID>) ]
111
112 rule ReceiveOfferService:
113 let
114     client = <$ClientID, $C_Endpoint>
115     server = <$ServiceID, $InstanceID, $S_Endpoint>
116 in
117 [   !C_Identity(client), In_P($C_Endpoint,
118     <'offer_service', server, nonce1>, nonce2)
119 ]
120 --[ E_C_ReceiveOfferService(client, server, nonce1),
121     OnlyOnce(<client, nonce2>),
122     Honest(client), Honest(server)
123 ]->
124 [   C_Available(client, server) ]
125
126 rule ReceiveStopService:
127 let
128     client = <$ClientID, $C_Endpoint>

```

---

```

129     server = <$ServiceID , $InstanceID , $S_Endpoint>
130 in
131 [ !C_Identity(client),
132   C_Available(client , server),
133   In_P($C_Endpoint, <'stop_service' , server , nonce1>, nonce2)
134 ]
135 --[ E_C_ReceiveStopOfferService(client , server ,
136   nonce1),
137   OnlyOnce(<client , nonce2>),
138   E_C_Available(client , server),
139   Honest(client), Honest(server)
140 ]->
141 []
142
143 // **** Request/Response Communication ****
144 // ** Server **
145 rule ReceiveRequest:
146 let
147     client = <$ClientID , SessionID , $C_Endpoint>
148     server = <$ServiceID , $InstanceID , $S_Endpoint>
149 in
150 [ !S_Identity(server),
151   S_State(server , 'open'),
152   In_P($S_Endpoint, <'request' , client ,
153     <$ServiceID , $S_Endpoint>, $MethodID ,
154     'request_data'>, nonce)
155 ]
156 --[ OnlyOnce(<server , nonce>),
157   E_S_OpenServiceOffer (server)
158 ]->
159 [ S_OpenRequest(server , client , $MethodID),
160   S_State(server , 'open')
161 ]
162
163 rule SendResponse:
164 let
165     client = <$ClientID , SessionID , $C_Endpoint>
166     server = <$ServiceID , $InstanceID , $S_Endpoint>
167 in
168 [ !S_Identity(server),
169   S_State(server , 'open'),
170   S_OpenRequest(server , client , $MethodID),

```

---

```

171     Fr(~nonce)
172 ]
173 --[ E_S_SendResponse($ServiceID , $ClientID ,
174     $C_Endpoint , SessionID , $MethodID ,
175     <'response_data' , ~nonce>),
176     E_S_OpenServiceOffer(server),
177     E_Send(<'response' ,<$ServiceID> , client , $MethodID , '
        response_data'>)
178 ]->
179 [   Out_P($C_Endpoint , <'response' , <$ServiceID> ,
180     client , $MethodID ,<'response_data' ,
181     ~nonce>>),
182     S_State(server , 'open')
183 ]
184
185 rule ReceiveFireAndForget:
186 let
187     client = <$ClientID , SessionID , $C_Endpoint>
188     server = <$ServiceID , $InstanceID , $S_Endpoint>
189 in
190 [   !S_Identity(server),
191     S_State(server , 'open'),
192     In_P($S_Endpoint , <'fire_and_forget' , client ,
193     <$ServiceID , $S_Endpoint> , $MethodID , data> ,
194     nonce)
195 ]
196 --[ E_S_ReceiveFireAndForget(server , <$ClientID ,
197     $C_Endpoint> , SessionID , $MethodID , data) ,
198     OnlyOnce(<server , nonce>),
199     E_S_OpenServiceOffer(server),
200     Honest(client) , Honest(server)
201 ]->
202 [   S_State(server , 'open')
203 ]
204
205 // ** Client **
206 rule SendRequest:
207 let
208     client = <$ClientID , $C_Endpoint>
209     server = <$ServiceID , $InstanceID , $S_Endpoint>
210 in
211 [   !C_Identity(client),

```



---

```

212     C_Available(client , server),
213     Fr(~SessionID)
214 ]
215 --[ E_C_SendRequest(client , ~SessionID , <$ServiceID ,
216     $S_Endpoint>,$MethodID, 'request_data' ),
217     E_C_Available(client , server),
218     E_Send(<'request' , <$ClientID , ~SessionID ,
219     $C_Endpoint>,<$ServiceID,$S_Endpoint> ,
220     $MethodID, 'request_data'>)
221 ]->
222 [ Out_P($S_Endpoint , <'request' , <$ClientID ,
223     ~SessionID , $C_Endpoint> , <$ServiceID ,
224     $S_Endpoint> , $MethodID, 'request_data'>),
225     C_OpenRequest(client ,<$ServiceID , $S_Endpoint> , $MethodID , ~
226     SessionID),
227     C_Available(client , server)
228 ]
229 rule ReceiveResponse :
230 let
231     client = <$ClientID , $C_Endpoint>
232     server = <$ServiceID , $InstanceID , $S_Endpoint>
233 in
234 [ !C_Identity(client),
235     C_OpenRequest(client , <$ServiceID , $S_Endpoint> , $MethodID ,
236     SessionID),
237     In_P($C_Endpoint , <'response' , <$ServiceID> ,
238     <$ClientID , SessionID , $C_Endpoint> ,
239     $MethodID, data> , nonce)
240 ]
241 --[ E_C_ReceiveResponse($ClientID , $C_Endpoint ,
242     $ServiceID , SessionID , $MethodID , data) ,
243     OnlyOnce(<client , nonce>)
244 ]->
245 [ ]
246 rule SendFireAndForget :
247 let
248     client = <$ClientID , $C_Endpoint>
249     server = <$ServiceID , $InstanceID , $S_Endpoint>
250 in
251 [ !C_Identity(client),

```

---

```

252     C_Available(client , server) ,
253     Fr(~SessionID)
254 ]
255 --[ E_C_SendFireAndForget(client , server ,
256     ~SessionID , $MethodID , 'request_data' ) ,
257     E_C_Available(client , server) ,
258     E_Send(<'fire_and_forget' , <$ClientID ,
259         ~SessionID , $C_Endpoint> ,
260         <$ServiceID , $S_Endpoint> ,
261         $MethodID , 'request_data' >)
262 ]->
263 [ Out_P($S_Endpoint , <'fire_and_forget' ,
264     <$ClientID , ~SessionID , $C_Endpoint> ,
265     <$ServiceID , $S_Endpoint> ,
266     $MethodID , 'request_data' >) ,
267     C_Available(client , server)
268 ]
269
270 // **** Corruption Rules ****
271 rule CorruptClient :
272 let
273     client = <$ClientID , $C_Endpoint>
274 in
275 [ !C_Identity(client) ]
276 --[ E_Corrupted(client) ,
277     OnlyOnce('corrupt')
278 ]->
279 [ !CorruptedChannel($C_Endpoint) ]
280
281 rule CorruptServer :
282 let
283     server = <$ServiceID , $InstanceID , $S_Endpoint>
284 in
285 [ !S_Identity(server) ]
286 --[ E_Corrupted(server) ,
287     OnlyOnce('corrupt')
288 ]->
289 [ !CorruptedChannel($S_Endpoint) ]
290
291 rule CorruptedChannelWrite :
292 [ In(<$Dst_Endpoint , x , nonce>) , !CorruptedChannel($Src_Endpoint) ]
293 -->

```

---

```

294 [In_P($Dst_Endpoint , x, nonce) ]
295
296 rule CorruptedChannelRead:
297 [In_P($Endpoint , x, nonce), !CorruptedChannel($Endpoint) ]
298 -->
299 [ Out(<x,nonce>) ]
300
301 // **** Channel Rules ****
302 rule PrivateChannel:
303 [ Out_P($Endpoint , x), Fr(~nonce) ]
304 --[ E_Transmit(x) ]->
305 [ In_P($Endpoint , x, ~nonce) ]
306 rule PrivateChannelBroadcast:
307 [Out_P('broadcast' , x), Fr(~nonce) ]
308 --[ E_Transmit(x)]->
309 [ In_P($Endpoint , x, ~nonce) ]
310
311 // **** Restrictions ****
312 restriction OnlyOnce:
313 " All x #i #j. OnlyOnce(x) @i & OnlyOnce(x) @j
314   => #i = #j"
315
316 restriction OnlyOneEndpointPerService:
317 " All ClientID C_Endpoint ServiceID InstanceID InstanceID_2
318   S_Endpoint S_Endpoint_2 nonce1 nonce2 #i #j.
319   E_C_ReceiveOfferService(<ClientID ,C_Endpoint>,<ServiceID ,
320     InstanceID ,S_Endpoint>,nonce1) @i
321   & E_C_ReceiveOfferService(<ClientID ,C_Endpoint>,<ServiceID ,
322     InstanceID_2 ,S_Endpoint_2>,nonce2) @j
323   & i < j
324   => Ex nonce3 #k. E_C_ReceiveStopOfferService(<ClientID ,
325     C_Endpoint>,<ServiceID , InstanceID , S_Endpoint>,nonce3) @k & k <
326     j
327 "
328 restriction BroadcastAddress:
329 // No identity can have 'broadcast' as its endpoint
330 " All #i.
331   OnlyOnce(<'create_client' , 'broadcast'>) @i
332   => F
333 & All #j.
334   OnlyOnce(<'create_server' , 'broadcast'>) @j
335   => F

```

---

```

331 "
332
333 // **** Helper lemmas **
334 lemma Availability [sources]:
335 " All client server #i.
336   E_C_Available(client , server) @i
337   ==> (Ex nonce #j. E_C_ReceiveOfferService(client , server , nonce)
338         @j & j < i)
339
340 lemma OpenServiceOffer [sources]:
341 " All server #i.
342   E_S_OpenServiceOffer(server) @i
343   ==> (Ex nonce #j. E_S_SendOfferService(server , nonce) @j & j < i)
344
345 // **** Executability lemmas ****
346 lemma RequestResponseComm:
347 exists-trace
348 " Ex ClientID C_Endpoint ServiceID SessionID MethodID data #i.
349   E_C_ReceiveResponse(ClientID , C_Endpoint , ServiceID , SessionID ,
350     MethodID , data) @i
351   & not(Ex x #k. E_Corrupted(x) @k)
352
353 lemma FireAndForgetComm:
354 exists-trace
355 " Ex ClientID C_Endpoint ServiceID InstanceID S_Endpoint SessionID
356   MethodID data #i #j.
357   E_C_SendFireAndForget(<ClientID , C_Endpoint> ,
358     <ServiceID , InstanceID , S_Endpoint> ,
359     SessionID , MethodID , data) @i
360   & E_S_ReceiveFireAndForget(<ServiceID , InstanceID , S_Endpoint> , <
361     ClientID , C_Endpoint> , SessionID , MethodID , data) @j
362   & not(Ex x #k. E_Corrupted(x) @k)
363
364 // **** Security lemmas ****
365 // ** Message authentication without corruption **
366 lemma AlivenessOfferService_NC:
367 // If a client accepts a service offer , apparently from server S , S
368 // has previously been active.
369 " All server client nonce1 #i.

```

---

```

368     E_C_ReceiveOfferService(client , server , nonce1) @i
369     ==> (Ex #j . E_S_Created(server) @j)
370         | (Ex id #k . E_Corrupted(id) @k)
371 "
372 lemma AlivenessStopOfferService_NC:
373 // If a client accepts a stop service offer from server S, S has
374 // previously been active.
375 " All server client nonce1 #i .
376     E_C_ReceiveStopOfferService(client , server , nonce1) @i
377     ==> (Ex #j . E_S_Created(server) @j)
378         | (Ex id #k . E_Corrupted(id) @k)
379 "
380 lemma OneTimeGroupAgreementOfferService_NC:
381 // If a client C accepts a service offer from a server S, the exact
382 // same offer has been previously send by S and has not previously
383 // been accepted by C.
384 " All server client nonce #i .
385     E_C_ReceiveOfferService(client , server , nonce)
386     @i
387     ==> (Ex #j . E_S_SendOfferService(server , nonce)
388         @j
389         & j < i
390         & not (Ex #i2 . E_C_ReceiveOfferService(client , server , nonce) @
391             i2 & not(#i2 = #i)))
392         | (Ex id #r . E_Corrupted(id)@r)
393 "
394 lemma OneTimeGroupAgreementFireAndForget_NC:
395 // If a server S accepts a fire&forget message from a client C, the
396 // exact same message has been previously send by C and has not
397 // previously been accepted by S.
398 " All server client session method data #i .
399     E_S_ReceiveFireAndForget(server , client , session , method , data) @
400     i
401     ==> (Ex #j . E_C_SendFireAndForget(client , server , session , method
402         , data) @j
403         & j < i
404         & not (Ex #i2 . E_S_ReceiveFireAndForget(server , client ,
405             session , method , data) @i2 & not(#i2 = #i)))
406         | (Ex id #r . E_Corrupted(id)@r)
407 "
408 "
409 "
410 // ** Message authentication with corruption **

```

---

```

401 lemma AlivenessOfferService:
402 // If a client accepts a service offer, apparently from server S, S
    has previously been active.
403 " All server client nonce #i.
404     E_C_ReceiveOfferService(client, server, nonce) @i
405     ==> (Ex #j. E_S_Created(server) @j)
406         | (Ex id #r. E_Corrupted(id) @r & Honest(id) @i)
407 "
408 lemma AlivenessStopOfferService:
409 // If a client accepts a stop service offer from server S, S has
    previously been active.
410 " All server client nonce1 #i.
411     E_C_ReceiveStopOfferService(client, server, nonce1) @i
412     ==> (Ex #j. E_S_Created(server) @j)
413         | (Ex id #r. E_Corrupted(id)@r & Honest(id) @i)
414 "
415 lemma OneTimeGroupAgreementOfferService:
416 // If a client C accepts a service offer from a server S, the exact
    same offer has been previously send by S and has not previously
    been accepted by C.
417 " All server client nonce #i.
418     E_C_ReceiveOfferService(client, server, nonce) @i
419     ==> (Ex #j. E_S_SendOfferService(server, nonce) @j
420         & j < i
421         & not (Ex #i2. E_C_ReceiveOfferService(client, server, nonce)
422             @i2 & not(#i2 = #i)))
423         | (Ex id #r. E_Corrupted(id)@r & Honest(id) @i)
424 "
425 lemma OneTimeGroupAgreementFireAndForget:
426 // If a server S accepts a fire&forget message from a client C, the
    exact same message has been previously send by C and has not
    previously been accepted by S.
427 " All server client session method data #i.
428     E_S_ReceiveFireAndForget(server, client, session, method, data) @
        i
429     ==> (Ex #j. E_C_SendFireAndForget(client, server, session, method
430         , data) @j
431         & j < i
432         & not (Ex #i2. E_S_ReceiveFireAndForget(server, client,
433             session, method, data) @i2 & not(#i2 = #i)))
434         | (Ex id #r. E_Corrupted(id)@r & Honest(id) @i)
435 "

```

---

```

433
434 // ** Practical attack lemmas **
435 lemma NoMitM:
436 // The adversary cannot get in a MitM position, acting as client
    towards a server and as server towards a client, relaying data and
    potentially modifying parts of the data
437 " All ClientID C_Endpoint1 C_Endpoint2 ServiceID S_Endpoint SessionID
438     MethodID data1 data2 nonce #i #j #k.
439     E_C_SendRequest(<ClientID , C_Endpoint1>, SessionID ,
440         <ServiceID , S_Endpoint>, MethodID, 'request_data') @i
441     & E_S_SendResponse(ServiceID , ClientID , C_Endpoint2, SessionID ,
442         MethodID, <data1 , nonce>) @j
443     & E_C_ReceiveResponse(ClientID , C_Endpoint1, ServiceID , SessionID
444         ,
445         MethodID, <data2, nonce>) @k
446     & i < j & j < k
447     & not(Ex id #r. E_Corrupted(id)@r & Honest(id) @i)
448     ==> C_Endpoint1 = C_Endpoint2
449 "
450 lemma NoDeassociationAttack:
451 // The adversary cannot 'hijack' an already existing client/server
    association
452 " All ClientID C_Endpoint ServiceID InstanceID S_Endpoint nonce
453     S_Endpoint_3 SessionID MethodID data #i #j #l.
454     ( E_S_SendOfferService(<ServiceID , InstanceID , S_Endpoint>, nonce)
455         @i
456     & E_C_ReceiveOfferService(<ClientID , C_Endpoint>, <ServiceID ,
457         InstanceID , S_Endpoint>, nonce) @j
458     & i < j
459     & not(Ex nonce2 #k. E_S_SendStopOfferService(<ServiceID ,
460         InstanceID , S_Endpoint>, nonce2) @k & k < l )
461     & E_C_SendRequest(<ClientID , C_Endpoint>, SessionID , <ServiceID ,
462         S_Endpoint_3>,
463         MethodID, data) @l & j < l
464     & not(Ex id #m. E_Corrupted(id) @ m & m < j)
465     & not(Ex id #r. E_Corrupted(id)@r & Honest(id) @i)
466     )
467     ==>
468     S_Endpoint = S_Endpoint_3
469 "
470 end

```

---

## A.2.2. Tamarin model of SESO-RC

For our model of SESO-RC (Section 6.3.6), we focus on the security of the proposed mechanism and, thus, providing a more abstract model than for our SOME/IP evaluation. Freshness values have been abstracted by nonces.

Listing A.17: SESO-RC Model

```
1 theory SESORc
2 begin
3 builtins: diffie-hellman, signing, hashing
4 functions: inc/1, dec/1 // increase/decrease freshness values
5 equations: dec(inc(x))=x
6 /*
7 Public Variables
8   $A, $B - identity information, such as Client/Service ID,
           Instance ID and Endpoint
9   $id_o - service ID that an identity is allowed to offering
10  $id_r - service ID that an identity is allowed to requests
11
12 Facts
13   !Ca_priv_key('ca',~ca_priv) - private key of Certificate
           Authority (CA)
14   !Ca_pub_key('ca',ca_pub) - public key of CA
15   !Identity($A,~priv_key, pub_key, cert) - Entity's identity and
           certificate information
16   !Cert_store($A,$B,cert_b) - $A stored certificate cert_b for
           entity $B
17   !Dh($server,~server_dh_priv) - stores the static Diffie-Hellman
           key of $server
18   Session($A,$B,s_key) - $A has an open session with $B with
           session key s_key
19 */
20
21 // **** Protocol ****
22 rule Create_ca:
23 let
24   ca_pub = pk(~ca_priv)
25 in
26 [ Fr(~ca_priv) ]
27 --[ OnlyOnce('create_ca') ]->
28 [   !Ca_priv_key('ca', ~ca_priv),
29   !Ca_pub_key('ca', ca_pub),
```



---

```

30     Out(<'ca', ca_pub>)
31 ]
32
33 rule Create_identity:
34 let
35     pub_key = pk(~priv_key)
36     cert= <$A, pub_key, $id_o, $id_r,
37         sign(<$A, pub_key, $id_o, $id_r>, ~ca_priv)>
38 in
39 [ Fr(~priv_key), !Ca_priv_key('ca', ~ca_priv) ]
40 --[OnlyOnce(<'create_id', $A>) ]->
41 [ !Identity($A, ~priv_key, pub_key, cert) ]
42
43 rule Distribute_cert:
44 [ !Identity($A, ~priv_key_a, pub_key_a, cert_a) ]
45 -->
46 [ Out(<'cert_exchange', cert_a>) ]
47
48 rule Reveal_priv_key:
49 [ !Identity($A, ~priv_key, pub_key, cert) ]
50 --[ E_Corrupt_identity($A) ]->
51 [ Out(~priv_key) ]
52
53 rule Store_cert:
54 let
55     cert_b= <$B, pub_key_b, $id_o_b, $id_r_b, signature_b>
56 in
57 [ !Identity($A, ~priv_key_a, pub_key_a, cert_a),
58   !Ca_pub_key('ca', ca_pubkey),
59   In(<'cert_exchange', cert_b>)
60 ]
61 --[ Eq(verify(signature_b, <$B, pub_key_b, $id_o_b,
62     $id_r_b>, ca_pubkey), true),
63     Neq(cert_a, cert_b),
64     E_Cert_stored($A, $B)
65 ]->
66 [ !Cert_store($A, $B, cert_b) ]
67
68 rule Offer_service:
69 let
70     server_dh_pub = 'g' ^ ~server_dh_priv
71 in

```

---

```

72 [ Fr(~server_dh_priv),
73   Fr(~server_fresh),
74   !Identity($server, ~priv_key_server, pub_key_server, server_cert)
75 ]
76 --[ ]->
77 [ Out(<'offer_service', $server, ~server_fresh,
78     server_dh_pub, h(server_cert),
79     sign(<'offer_service', $server, ~server_fresh,
80     server_dh_pub, h(server_cert)>,
81     ~priv_key_server)>),
82   !Dh($server, ~server_dh_priv, ~server_fresh)
83 ]
84
85 rule Request_service:
86 let
87   client_dh_pub = 'g'^~client_dh_priv
88   server_cert= <$server, server_pub_key,
89     $server_id_o, $server_id_r,
90     sign_server_cert>
91   client_cert= <$client, client_pub_key,
92     $client_id_o, $client_id_r,
93     sign_client_cert>
94   session_key = server_dh_pub^~client_dh_priv
95   client_fresh = inc(server_fresh)
96 in
97 [ Fr(~client_dh_priv),
98   !Cert_store($client, $server, server_cert),
99   !Identity($client, ~priv_key_client, pub_key_client, client_cert)
100
101   In(<'offer_service', $server, server_fresh, server_dh_pub,
102     server_cert_hash, signed_offer>
103
104   ]
105 --[ Eq(verify(signed_offer, <'offer_service',
106   $server, server_fresh, server_dh_pub,
107   server_cert_hash>, server_pub_key), true),
108   Eq(h(server_cert), server_cert_hash),
109   Eq($server_id_o, $client_id_r),
110   E_C_Session_established($client, $server,
111   client_fresh, session_key),
112   Honest($client), Honest($server),
113   Secret(session_key)
114 ]->

```

---

```

112 [ Session($client , $server ,
113     server_dh_pub^~client_dh_priv),
114 Out(<'request_service' , $client , $server ,
115     client_fresh , client_dh_pub ,
116     h(client_cert),
117 sign(<'request_service' , $client , $server ,
118     client_fresh , client_dh_pub ,
119     h(client_cert)> , ~priv_key_client)>)
120 ]
121
122 rule Response_service_request:
123 let
124     server_cert = <$server , server_pub_key ,
125         $server_id_o , $server_id_r ,
126         sign_server_cert>
127     client_cert = <$client , client_pub_key ,
128         $client_id_o , $client_id_r ,
129         sign_client_cert>
130     session_key = client_dh_pub^~server_dh_priv
131     client_fresh = inc(~server_fresh)
132 in
133 [ In(<'request_service' , $client , $server ,
134     client_fresh , client_dh_pub ,
135     client_cert_hash , signed_request> ,
136     !Identity($server , ~priv_key_server , pub_key_server , server_cert) ,
137     !Cert_store($server , $client , client_cert) ,
138     !Dh($server , ~server_dh_priv , ~server_fresh)
139 ]
140 --[ Eq(verify(signed_request , <'request_service' ,
141     $client , $server , client_fresh ,
142     client_dh_pub , client_cert_hash> ,
143     client_pub_key) , true) ,
144     Eq(h(client_cert) , client_cert_hash) ,
145     Eq($server_id_o , $client_id_r) ,
146     E_S_Session_established($server , $client , client_fresh , session_key)
147     ,
148     OnlyOnce(<$server , ~server_fresh , client_dh_pub>) ,
149     Honest($client) , Honest($server) ,
150     Secret(session_key)
151 ]->
152 [ Session($server , $client , client_dh_pub^~server_dh_priv)
153 ]

```

---

```

153
154 // **** Restrictions ****
155 restriction Equality:
156   "All x y #i. Eq(x,y) @i ==> x = y"
157
158 restriction OnlyOnce:
159   "All #i #j x. OnlyOnce(x)@#i & OnlyOnce(x)@#j ==> #i = #j"
160
161 restriction Inequality:
162   "All x #i. Neq(x,x) @ #i ==> F"
163
164 // **** Executability lemmas ****
165 lemma Certificates_can_be_exchanged:
166   exists-trace
167   // Certificates can be exchanged without any corrupted parties
168   "Ex server client #i.
169   (   E_Cert_stored(server, client) @i
170     & not(Ex #k. E_Corrupt_identity(server) @k)
171     & not(Ex #k. E_Corrupt_identity(client) @k)
172   )"
173
174 lemma Session_can_be_established:
175   // A session can be established without any corrupted parties
176   exists-trace
177   "Ex server client session_key session_id #i #j.
178   (   E_S_Session_established(server, client,
179     session_id, session_key) @i
180     & E_C_Session_established(client, server,
181     session_id, session_key) @j
182     & j < i
183     & not(Ex #k. E_Corrupt_identity(server) @k)
184     & not(Ex #k. E_Corrupt_identity(client) @k)
185   )"
186
187 // **** Security lemmas ****
188 lemma Perfect_forward_secretcy:
189   // The adversary cannot learn a session key of a session between a
190     client C and a server S without having corrupted either C or S
191     before the key exchange.
192   "All x #i. Secret(x) @i
193     ==>
194     not (Ex #j. K(x) @j)

```

---

```

193     | (Ex id #r. E_Corrupt_identity(id) @r
194         & Honest(id) @i & r < i)
195 "
196
197 lemma Weak_agreement:
198 // If a server S established a session, apparently with a client C,
199   C also established a session, apparently with S.
200 "All server client session_id session_key #i.
201   E_S_Session_established(server, client, session_id, session_key)
202     @i
203   =>
204   (Ex session_id2 session_key2 #j.
205     E_C_Session_established(client, server, session_id2,
206       session_key2) @j)
207   | (Ex id #r. E_Corrupt_identity(id) @r
208     & Honest(id) @i)
209 "
210
211 lemma Non_injective_agreement:
212 // If a server S established a session, apparently with a client C,
213   C established a session, apparently with S, and they agree on all
214   session parameters.
215 "All server client session_id session_key #i.
216   E_S_Session_established(server, client, session_id, session_key)
217     @i
218   =>
219   (Ex #j. E_C_Session_established(client, server, session_id,
220     session_key) @j)
221   | (Ex id #r. E_Corrupt_identity(id) @r
222     & Honest(id) @i)
223 "
224
225 lemma Injective_agreement:
226 // If a server S established a session, apparently with a client C,
227   C established a session, apparently with S, and they agree on all
228   session parameters. Each session of S corresponds to a unique
229   session of C.
230 "All server client session_id session_key #i.
231   E_S_Session_established(server, client, session_id, session_key)
232     @i
233   =>

```

---

```

224     (Ex #j . E_C_Session_established(client , server ,
225         session_id , session_key) @j
226     & j < i
227     & not (Ex server2 client2 #i2 . E_S_Session_established(
228         server2 ,
229         client2 , session_id , session_key)
230         @i2
231     & not (#i2 = #i)))
231 | (Ex id #r . E_Corrupt_identity(id) @r
232     & Honest(id) @i)
233 ”
234 end

```

### A.2.3. Tamarin model of SESO-AS

In our model of SESO-AS, freshness values have been abstracted by nonces. Moreover, the broadcast of service offer messages by the AS is represented by individual messages per client.

Listing A.18: SESO-AS Model

```

1  theory SESOas
2  begin
3
4  builtins: symmetric-encryption , hashing
5  functions: mac/2
6  // Computes a message authentication code as mac(data , key)
7  /*
8  Public Variables
9     $A, $S, $C – Identities. $S represents a server, $C a client
10
11 Variables
12     os – service offer
13     f0, f1 – freshness values (abstracted by nonces)
14     kS, kC – symmetric keys shared between Authentication Server (AS)
15             and a server or client
16     kSe – session key of a server , masterkey to derive shared client
17           keys
18     kSC – symmetric key shared between a server and a client

```

---

```

19 Facts
20     !Identity($A, ~kA) – Identity and corresponding key shared with
        the AS
21     !Shk($C, $S, kSC) – kSC is believed a shared key between $C and $
        S by $C
22 */
23
24 // **** Protokol ****
25 rule Create_identity:
26     [Fr(~kA)]
27     -->
28     [!Identity($A, ~kA)]
29
30 rule Corrupt_identity:
31     [!Identity($A, ~kA)]
32     --[ Reveal($A) ]->
33     [Out(~kA)]
34
35 rule OfferServiceSAS:
36     let
37         os = <$S, ~data, ~f0>
38         e0 = <$S, senc(~kSe, kS)>
39     in
40     [ !Identity($S, kS),
41       Fr(~kSe), Fr(~f0), Fr(~data)
42     ]
43     --[ E_S_Send($S, os, ~kSe),
44         Honest($S),
45         Secret1(~kSe)
46     ]->
47     [Out(<os, e0, mac(<os, e0>, kS)>)]
48
49 rule OfferServiceASC:
50     let
51         os_in  = <$S, data, f0>
52         e0     = <$S, senc(kSe, kS)>
53         os_out = <$S, data, ~f1>
54         kSC    = h(kSe, $C)
55         e1     = <$C, senc(kSC, kC)>
56     in
57     [ !Identity($S, kS),
58       !Identity($C, kC),

```

---

```

59     Fr(~f1),
60     In(<os_in, e0, mac(<os_in, e0>, kS)>)
61   ]
62   --[ OnlyOnce(<'offer_service_as', f0>),
63     OnlyOnce(<'offer_service_as', ~f1>),
64     E_AS_Receive($S, os_in, kSe),
65     E_AS_Send($S, $C, os_out, kSC)
66   ]->
67   [ Out(<os_out, e1, mac(<os_out, e1>, kC)>) ]
68
69 rule ReceiveOfferService:
70   let
71     os = <$S, data, f1>
72     e1 = <$C, senc(kSC, kC)>
73   in
74   [ !Identity($C, kC),
75     In(<os, e1, mac(<os, e1>, kC)>)
76   ]
77   --[ E_C_Receive($C, os, kSC),
78     OnlyOnce(<'offer_service_c', f1>),
79     Honest($C),
80     Honest($S),
81     Secret2(kSC),
82     NotEq($C, $S)
83   ]->
84   [!Shk($C, $S, kSC)]
85
86 // **** Restrictions ****
87 restriction OnlyOnce:
88 "All x #i #j. OnlyOnce(x) @i & OnlyOnce(x) @j
89  => #i = #j"
90
91 restriction NotEq:
92 "All x y #i. NotEq(x, y) @i => not(x = y)"
93
94 // **** Executability lemmas ****
95 lemma Key_can_be_exchanged:
96 exists-trace
97 "Ex server client data f0 f1 key0 key1 #i #j.
98   E_S_Send(server, <server, data, f0>, key0) @i
99   & E_C_Receive(client, <server, data, f1>, key1) @j
100  & i < j"

```



---

```

101     & not(Ex id #r. Reveal(id) @r)
102  "
103
104 // **** Security lemmas ****
105 lemma Non_injective_agreement:
106 //   If a client C receives a message, apparently from a server S, S
107 //   has previously sent this message to C and both agree on the
108 //   content.
109 "All server client data f1 kSC #i.
110   E_C_Receive(client, <server, data, f1>, kSC) @i
111   => (Ex f0 kSe #j. E_S_Send(server, <server, data, f0>, kSe) @j &
112     kSC = h(kSe, client))
113     | (Ex #r1. Reveal(server) @r1)
114     | (Ex #r2. Reveal(client) @r2)
115  "
116 lemma Injective_agreement:
117 //   If a client C receives a message, apparently from a server S, S
118 //   has previously sent this message to C and both agree on the
119 //   content.
120 //   In addition, the receive event corresponds to a unique send event
121 //   .
122 "All server client data f1 kSC #i.
123   E_C_Receive(client, <server, data, f1>, kSC) @i
124   => ( (Ex f0 kSe #j. E_S_Send(server, <server, data, f0>, kSe)
125     @j
126     & j < i
127     & kSC = h(kSe, client))
128     & not (Ex server2 data2 f2 kSe #k. E_C_Receive(client, <
129     server2, data2, f2>, kSC) @k
130     & kSC = h(kSe, client)
131     & not (#k = #i))
132   )
133   | (Ex #r1. Reveal(server) @r1)
134   | (Ex #r2. Reveal(client) @r2)
135  "
136 // ** Secrecy **
137 lemma Secrecy_of_session_key:
138 //   The adversary cannot learn a session key generated by a server S
139 //   without computing S.
140 "All x #i.
141   Secret1(x) @i
142   => not(Ex #j. K(x) @j)

```

---

```

134         | (Ex id #r. Reveal(id) @r & Honest(id) @i)
135     ”
136 lemma Secrecy_of_shared_symmetric_key:
137 // The adversary cannot learn a shared symmetric key between a
138     client C and a server S without corrrpting either C or S.
139 ”All x #i.
140     Secret2(x) @i
141     => not(Ex #j. K(x) @j)
142     | (Ex id #r. Reveal(id) @r & Honest(id) @i)
143 ”
144 // ** Perfect Forward Secrecy **
145 lemma PFS_of_session_key:
146 // The adversary cannot learn a session key generated by a server S
147     which has not previously been corrrputed.
148 ”All x #i.
149     Secret1(x) @i
150     => not(Ex #j. K(x) @j)
151     | (Ex id #r. Reveal(id) @r & Honest(id) @i & r < i)
152 ”
153 lemma PFS_of_shared_symmetric_key:
154 // The adversary cannot learn a shared symmertic key between a
155     client C and a server S without either of these parties being
156     corrrupted before the key exchange occurs.
157 ”All x #i.
158     Secret2(x) @i
159     => not(Ex #j. K(x) @j)
160     | (Ex id #r. Reveal(id) @r & Honest(id) @i & r < i)
161 ”
162 end

```