Model-Driven Security Test Case Generation Using Threat Modeling and Automata Learning

Stefan Marksteiner
MODEL-DRIVEN SECURITY TEST CASE GENERATION USING THREAT MODELING AND AUTOMATA LEARNING

Stefan Marksteiner

2024

School of Innovation, Design and Engineering
Abstract

Automotive systems are not only becoming more open through developments like advanced driving assistance functions, autonomous driving, vehicle-to-everything communication and software-defined vehicle functionality, but also more complex. At the same time, technology from standard IT systems become frequently adopted in these systems. This development has two negative effects on correctness and security: the rising complexity adds potential flaws and vulnerabilities while the increased openness expands attack surfaces and entry points for adversaries.

To provide more secure systems, the amount of verification through testing has to be significantly increased, which is also a requirement by international regulation and standards. Due to long supply chains and non-disclosure policies, verification often have to use black-box methods. This thesis strives therefore towards finding more efficient methods of automating test case generation in both white- and black-box scenarios. The thesis focuses on communication protocols used in vehicular systems and we base our research on model-based methods. Our contributions include:

• We provide a practical method to automatically obtain behavioral models in the form of state machines of communication protocol implementations in real-world settings using automata learning.

• We also provide a means to automatically check these implementation models for their compliance with a specification (e.g., from a standard).

• We furthermore present a technique to automatically derive test-cases to demonstrate found deviations on the actual system.

• We also present a method to create abstract cybersecurity test case specifications from semi-formal threat models using attack trees.
Sammanfattning

Fordonssystem blir idag mer och mer öppna och komplexa genom introduktion av t.ex. avancerade körhjälpfunctioner, autonom körning, kommunikation mellan fordon och allt mer mjukvarudefinierad fordonsfunktionalitet. Samtidigt används teknik från traditionella IT-system allt oftare i dessa system. Denna utveckling har två negativa effekter på korrekthet och säkerhet: den ökade komplexiteten lägger till potentiella brister och sårbarheter medan den ökade öppenheten utökar attackytor och ingångspunkter för digitala inkräktare.

För att tillhandahålla säkrare system måste mängden verifiering av systemsäkerhet genom testning ökas avsevärt, vilket också är ett krav enligt internationella regler och standarder. På grund av långa värdekedjor och sekretessprinciper måste verifieringsmetoder ofta använda s.k. black-box verifiering där man verifierar funktionen av en komponent utan att kunna studera hur den fungerar internt. Det är dock fördelaktigt om man kan använda s.k. while-box metoder där man kan verifiera en komponent medan man samtidigt studerar hur den fungerar internt.

To my family
Acknowledgments

I like to use this space to thank this thesis’ advisors Marjan Sirjani and Mikael Sjödin for spending their precious time and giving me guidance in research directions and writing, which brought me forward in both professional and scientific manners. Furthermore, I like to thank my employer at the AVL List GmbH, foremost my former and current supervisors Horst Pflügl and Kieran McAleer for giving me the freedom to do the necessary research, without losing the vision for the application and the practical use of it. I also want to thank my son, Benjamin Marksteiner, just for being there and bringing joy (and also action) to my life. Most of all, I like to thank my wife Astrid Marksteiner for going this journey with me, even if it is not easy alongside manyfold job duties and a small kid. No acknowledgement could make it up with her support.

Stefan Marksteiner, February 2024, Graz, Austria

This research received funding from the program “ICT of the Future” of the Austrian Research Promotion Agency (FFG) and the Austrian Ministry for Transport, Innovation and Technology under grant agreements No. 867558 (project TRUSTED) and No. 880852 (project LEARNTWINS) and within the ECSEL Joint Undertaking (JU) under grant agreements No. 876038 (project InSecTT) and No. 101007350 (project AIDOaRt). The JU receives support from the European Union’s Horizon 2020 research and innovation programme and Austria, Sweden, Spain, Italy, France, Portugal, Ireland, Finland, Slovenia, Poland, Netherlands, Turkey. The document reflects only the author’s view and the Commission is not responsible for any use that may be made of the information it contains.
List of Publications

Papers Included in the Licentiate Thesis


---

1The included papers have been reformatted to comply with the thesis layout.
Additional Peer-reviewed Publications Related to the Thesis


These papers are not included in this thesis.
Contents

I Thesis 1

1 Introduction 3
   1.1 Practical Motivation: Industrial Problems Leading to Scientific Problems 5
   1.2 Thesis Outline 7

2 Background and Preliminaries 9
   2.1 Threat Modeling 9
   2.2 Attack Trees 10
   2.3 Labelled Transition Systems 10
   2.4 Formalized Attack Languages 10
   2.5 Mealy Machines 11
   2.6 Automata Learning 11
   2.7 Behavioral Equivalences 12

3 Research Overview 13
   3.1 Research Goals 14
      3.1.1 Threat Model-based Test Generation 16
      3.1.2 Automated State Machine Derivation 16
      3.1.3 Compliance Checking 17
   3.2 Research Method 18

4 Research Contributions 21
   4.1 Threat Model-based Test Generation 23
   4.2 Automated State Machine Derivation 23
   4.3 Compliance Checking 24
   4.4 Publications 25
4.4.1 Paper I ................................................. 25
4.4.2 Paper II ............................................... 26
4.4.3 Paper III ............................................... 26
4.4.4 Paper IV ................................................ 27
4.4.5 Related Papers Not Included in the Thesis ............... 28

5 Related Work .............................................. 31
  5.1 Model-based Test Case Generation ......................... 31
  5.2 Attack Trees-based Security Testing ....................... 32
  5.3 Formalized Test Descriptions ............................. 32
  5.4 Automated State Machine Derivation and Protocol Learning . 33
  5.5 Conformance Checking using Equivalence of State Machines . 34

6 Conclusion and Future Work .............................. 35
  6.1 Conclusions ............................................ 35
  6.2 Future Directions ....................................... 36

Bibliography ................................................... 39

II Included Papers ............................................. 47

7 Paper I: A Systematic Approach to Automotive Security .... 49
  7.1 Introduction ............................................. 51
  7.2 TRUSTED Methodology .................................. 52
  7.3 Automotive Security by Design .......................... 54
    7.3.1 System Architecture Model ......................... 55
    7.3.2 Threat Analysis ..................................... 55
    7.3.3 V&V Planning ....................................... 57
  7.4 Automotive Security Testing ............................. 58
    7.4.1 Automata Learning for Correctness ................. 58
    7.4.2 Use-Case Scenarios ................................. 59
  7.5 Conclusion ............................................... 62
Bibliography .................................................. 63

8 Paper II: Approaches For Automating Cybersecurity Testing Of Connected Vehicles 67
  8.1 Introduction ............................................. 69
8.2 State of the Art and Related Work .......................... 70
8.3 Automotive Cybersecurity Lifecycle Management .......... 72
  8.3.1 Threat Modeling ........................................... 73
8.4 Cybersecurity Testing ........................................ 74
  8.4.1 Learning-based Testing ................................... 76
  8.4.2 Model-based Test Case Generation ....................... 78
  8.4.3 Testing Platform ......................................... 79
  8.4.4 Automated Test Execution ............................... 80
  8.4.5 Fuzzing .................................................. 82
8.5 Conclusion .................................................. 83
Bibliography ...................................................... 85

9 Paper III:
Using Automata Learning for Compliance Evaluation of Communication Protocols on an NFC Handshake Example 91
9.1 Introduction .................................................. 93
  9.1.1 Motivation ................................................ 93
  9.1.2 Contribution ............................................. 94
9.2 Preliminaries ................................................ 94
  9.2.1 State Machines .......................................... 95
  9.2.2 Transitions and Equivalence ............................ 95
  9.2.3 Automata Learning ...................................... 98
  9.2.4 LearnLib ................................................ 99
  9.2.5 Near Field Communication .............................. 100
  9.2.6 The NFC Handshake Automaton ....................... 100
9.3 NFC Interface ............................................... 103
  9.3.1 Learner Interface Device ............................... 103
  9.3.2 Adapter Class ........................................... 103
9.4 Learning Setup .............................................. 104
  9.4.1 Comparing Learning Algorithms and Calibrations .... 105
  9.4.2 Abstraction ............................................. 107
  9.4.3 Labeling and Simplification ........................... 109
  9.4.4 Compliance Evaluation ................................. 109
9.5 Evaluation .................................................. 111
  9.5.1 Test Cards and Credit Cards ........................... 111
  9.5.2 Passports ............................................... 111
  9.5.3 Tesla Key Fob .......................................... 111
9.6 Related Work ................................................ 113
9.7 Conclusion .................................................. 113
9.7.1 Discussion ............................................. 113
9.7.2 Outlook .............................................. 114
Bibliography .............................................. 115

10 Paper IV: From TARA to Test: Automated Automotive Cybersecurity Test Generation Out of Threat Modeling 119
10.1 Introduction ........................................... 121
  10.1.1 Motivation ......................................... 122
  10.1.2 Contribution ...................................... 122
10.2 Automotive Security Communication ...................... 123
  10.2.1 Cybersecurity Assurance Level (CAL) .............. 123
  10.2.2 Target Attack Feasibility (TAF) .................... 124
  10.2.3 Integrating CAL and TAF in security testing ....... 124
10.3 Threat Modeling ....................................... 128
  10.3.1 Threat-Interdependencies and Attack Trees ....... 130
10.4 Automated Testing .................................... 131
  10.4.1 Security Tests and their relationship with the Security Analysis ...................... 132
  10.4.2 Security Test Generation ......................... 132
10.5 Case Study .......................................... 135
10.6 Related Work ....................................... 136
10.7 Conclusion .......................................... 138
Bibliography .............................................. 139
I

Thesis
Chapter 1

Introduction

This thesis concentrates on using formal methods for assuring correctness and security of systems with a focus on the automotive domain. To utilize these methodologies in practice, some practical problems have to be solved, that in turn lead to solving some scientific problems that lay the fundament for the provided solutions. A special emphasis is given to verifying the correctness of communication protocol implementations in system components and to verification from an architectural threat modeling perspective. The upcoming UN-ECE regulation R.155 [7] mandates not only the introduction of a cybersecurity management system (CSMS) and according security measures for automotive systems, but also evidence of their appropriateness and effectiveness, which is to be demonstrated by testing. This requires an amount of testing of the cybersecurity of vehicle systems that is not to be covered with manual testing alone. Therefore, automated methodologies for automotive cybersecurity testing are needed. Furthermore, the testing should work in both white box and black box settings. Methods for the former do not work on the latter (for the lack of information), however black box methods are not efficient in white box settings (for not using present information). The former to cover the security of all aspects of a system in the most thorough way. The latter to on the one hand provide an attacker’s view but also because of the lack of access to source code or other system component internals due to long supply chains, as well as the unwillingness of manufacturers to disclose system details.

Formal methods, particularly formal modeling has been used in engineering complex systems, for example in the automotive and aerospace domains. Despite the effort they normally come with, their founding on strong mathe-
Chapter 1. Introduction

Mathematical principles has three main advantages: a very structured approach that provides a good level of comprehensiveness, well-reasoned verdicts, and a high degree of automation capability (once a model has been derived) due to computer systems basing on the same logic. There are different approaches to formal modeling practically in use in the industry: models based on state machines have a long tradition of analyzing systems’ correctness, on the other hand architectural threat models [8] in the form of graphs with clear semantics and first order logic-based threat rule sets are used in the software industry for quite some time and have become very widely adopted as part of a Threat Analysis and Risk Assessment (TARA) in the automotive industry [9]. Threat models analyze a system design based on data flows between its architecture’s components [8].

Regardless of these beneficial traits, these methods are not only ordinarily very labor-intense, but also hard to apply on black-box components, as without access to internal information it is not trivial to obtain a state machine model (also called automaton) in an automated way. Even if such an automaton is present, reliable methods and rules to check it for correctness in terms of security have to be in place – none such is so far formulated in general, therefore sets of these rules and methods have to be established for each use case individually. Lastly, it is an open question how white box and black box modeling methods could support each other. An architectural threat model from a TARA is usually (manually) generated at design time and is based on assumptions that are often tacitly made when using modeling components. We therefore strive to investigate, if it is possible to check whether made assumptions actually hold in the implementation. Concentrating on external interfaces of system components, automatically derived state machine models of communication protocol are investigated. One further question is also which algorithm and parameter configuration is most suitable for inferring models of implementations of a specific protocol.

This thesis therefore provides an approach to generate security test cases from an architectural threat model in white box scenarios, concentrating on the question how to provide a formal translation from threat modeling results to actions in test cases utilizing attack trees [10, 11] and labelled transition systems (LTS) [12]. For black box scenarios, this thesis investigates inferring a behavioral model in form of a state machine. To derive such a model, the L algorithm by Angluin [13] and variations thereof [14], as well as more modern algorithms (e.g. [15]) were examined. Automatic model derivation is beneficial because not only manual modeling in general is very resource-intense, but also hard to perform at all in black box settings. On an example case of the
Near-field Communication’s (NFC) handshake protocol (ISO 14443-3 [16]), for which a learning interface working in real-world environments is provided, the different algorithm and parameter sets were investigated, giving details (efficacy and performance) for automata learning in special cases. The thesis also works out the necessary level of abstraction in order to investigate the possibilities of implementing this concept in a practical (i.e., real-world) setup. Using these learned models, a behavior-based black box compliance checking method using bisimulation or trace equivalence was provided. The comparison object for these equivalences is an automaton modeled after the respective specification or standard, the learned implementation should comply to. The reason for choosing bisimulation at this point, despite trace equivalence being the actual property for black box compliance, is its superior efficiency over the latter. The threat model-based test case generation is a novel approach, while the method of combining automata learning and bisimulation checking is rarely used in general, and not at all for behavior-based black box implementation compliance checking. Nevertheless, it provides a more rigorous way of compliance checking due to its more exhaustive way of model building. So far, little comprehensive work for automata learning in different settings has been performed, for which this thesis also provides a contribution. Lastly, the thesis sketches some notions of using feedback from the behavioral state machine checking to the architectural threat models. The latter are ordinarily generated manually during design phase and based on assumptions about the modeled components (e.g., a modeled component complying to a certain standard or featuring a distinct property). These assumptions can be checked in later phases by checking the state machines of the implemented systems if these assumptions hold (e.g., if the implemented component actually complies to the standard). This allows for statements about the accuracy of the threat model from checking the behavioral state machine models.

1.1 Practical Motivation: Industrial Problems Leading to Scientific Problems

This section contains an outline of what practical problems motivate the research goals (Section 3.1) and solutions to scientific problems (Section 4) in this thesis. It also serves the purpose of giving a context how the problem fields for the research goals are related, targeting to give a better understanding of the overall picture of the research. The practical overall motivation of this thesis is facilitating automated cybersecurity testing of vehicular systems.
Chapter 1. Introduction

Analyzing contemporary state-of-the-art automotive cybersecurity engineering processes (most prominently ISO/SAE 21434:2021 [17]), several gaps that hinder the efficiency of testing were identified. A standard cybersecurity engineering process is aligned with the general automotive engineering processes. One proliferated example is Automotive Spice, which is roughly defined along four major activities [18]:

- Threat Analysis and Risk Assessment (TARA – accompanying the system design as part of a cybersecurity security requirements elicitation process) to analyze potential weaknesses and threats in the design and assess their severity during the design phase.

- Implement the system using security goals and claims drawn from the TARA mitigating the found risks and therefore implementing a secure design (in cybersecurity implementation process).

- Validate and verify the system security measurements’ effectiveness, providing evidence and arguments for the implemented system’s cybersecurity (as risk treatment verification and validation processes).

- Repeat the actions above during the rest of the system’s life cycle after the start of production (SOP) with both updates of the system (functional and non-functional) and of the threat landscape (e.g., discovery of new vulnerabilities).

To provide both more efficient and more rigorous testing of system correctness from a cybersecurity perspective, structuring and automation is desirable to be applied. One of potential fields identified is to combine threat modeling done in the TARA with testing by automatically deriving test cases from the former. This enhances efficiency by doing two necessary things at once (TARA and test case definition) and enhances testing rigor by directing the testing to the very security measures derived from the security goals drawn from the TARA. To gain this ability, formalized test cases descriptions (done in previous work) has to be connected to the results of a TARA, constituting a research problem resulting in research goal 1 (RG1 – Section 3.1.1). Although the TARA is conducted during design time (without an implementation available), it is still possible to create the necessary test case descriptions, as those can be made technology-agnostic and formed into practically executable test cases once an implementation is available (see Section 5.3). Also, it is quite usual in the automotive industry that an Original Equipment Manufacturer (OEM – in the automotive context mostly a car maker) integrates components from suppliers
without getting access to its internals (i.e., not getting source code, internal specifications, etc.), engineered after the OEM’s specifications. This creates the needs to test the correctness and security of these systems in a black box setting. One scientific problem this requirement opens up is a means to automatically obtain a formalized description of the behavior of a (sub-)system’s implementation to have an object to automatically analyze, which eventually lead to research goal 2 (RG2 – Section 3.1.2) how to create a state machine model from a black box system. The other side of the medal is to have a means to actually check that model for its correctness and security. This relays to the research problem of how to check the behavior of a model against a given specification, yielding research goal 3 (RG3 – Section 3.1.3). Both the solution for RG1 and for RG3 have been created after industrial needs which is underpinned by patents that have been filed (Austrian patent applications No. A50667/2023 and A50660/2023, respectively). Future work after the licenti- ate completion includes methods to use model checking for dedicated security properties, use learnt models for fuzz testing and derive features to check from threat modeling in order to check modeled assumptions more rigorously. As this kind of test automation usually iterate over the complete life cycle the correct- ness of the implementation also feeds back to the threat modeling, by check if the assumptions made in the design phase hold in the implementation, which influences future iterations in the life cycle. Figure 1.1 gives an overview of the automotive cybersecurity process group [18] and the practical implications of its automation leading to the research goals of this thesis; the boxes represent the processes of the cybersecurity engineering process group, while the arrows represent practical improvements as presented in this thesis (see above), leading to the research goals.

1.2 Thesis Outline

This thesis consists of two parts: Part I provides a coat for the research, namely the necessary preliminaries, the aim of the research, its contribution and comparison with existing work. Part II consists of the research papers constituting this thesis. the remainder of Part I is organized as follows: Chapter 2 contains the background, Chapter 3 gives and overview of the research goals and methods, Chapter 4 outlines the research contributions including a short description of the included publications, Chapter 5 outlines related work and Chapter 6 concludes the thesis and gives an outlook to future work.
Figure 1.1: Research contributions in relation to the Security Engineering Process after Automotive Spice [18]. The boxes represent the processes of the cybersecurity engineering process group, while the arrows represent practical improvements as presented in this thesis (see above), leading to the research questions in Section 3.1.
Chapter 2

Background and Preliminaries

This section very briefly explains some basic concepts that are used in this thesis. Other related work and alternative approaches towards reaching the research goal are outlined in Section 5. The usage of this background research this thesis builds on is as follows: Threat Modeling, Attack Trees, Labelled Transition Systems (LTS'), and Formalized Attack Languages in RG1 (Section 3.1.1), Mealy Machines and Automata Learning in RG2 (Section 3.1.2), Mealy Machines and Behavioral Equivalences in RG3 (Section 3.1.3).

2.1 Threat Modeling

Threat modeling is a systematic, semi-formal approach to scrutinize systems for potential threats and pitfalls. Threat modeling ordinarily requires two components [8]. One is a structured representation of the considered system (i.e., a system model), containing all information necessary to determine potential threats, as well as assessing their impact and likelihood of occurrence. A commonly used form of representation in sophisticated tools for threat modeling are data flow diagrams. The second component is the actual threat model. This model consists of a set of rules that determine which potential threat would occur if two components in the system model are connected in a certain way. These rule sets can, dependent on the application domain, become very complex, with the goal being to scrutinize the considered system very compre-
Chapter 2. Background and Preliminaries

hensively and structured. Sophisticated threat models contain a considerable amount of domain knowledge and are usually created by groups of security experts in the respective domain. This thesis uses threat models as a basis to create test cases in a structured way (see Section 4.1).

2.2 Attack Trees

Attack trees display relations, interdependencies and hierarchies of threats and vulnerabilities [10, 11]. The advantage of this form of representation is the ability to display different paths towards a certain objective i.e., to show different opportunities to concatenate attacks in order to exploit a certain vulnerability from a distinct starting point (mostly an interface accessible from the outside). This way, attack trees are a capable tool for assessing how combined attacks that exploit a complete set of threats impact the overall attack surface and success likelihood [19]. In this thesis, attack trees stemming from threat models are the origin for a method to automatically derive technology-agnostic security test scenarios to provide evidence for the correct functioning of implemented security measures (see Section 4.1).

2.3 Labelled Transition Systems

A principal notation for formal representations of systems used in this thesis are Transition Systems (TS) and Labelled Transition System (LTS). A TS is defined as a set of states (Q) and a transition relation (→ ∈ Q × Q, with q, q′ ∈ Q; q → q′). A Labelled Transition System (LTS) additionally possesses a set of labels (Σ), such that each transition is named with a label σ in Σ (q, q′ ∈ Q, σ ∈ Σ; q → q′) [12]. LTS can describe the behavior of systems and mechanisms at different levels. This thesis uses LTS for a translation mechanism from attack trees to attack descriptions in a specifically designed attack description language (see Section 4.1).

2.4 Formalized Attack Languages

Domain-specific languages (DSLs) are computer programming language of limited expressiveness focused on a particular domain [20]. That means that they should be just expressive enough to model any necessary features and conditions of the respective domain and lean enough for domain experts to be
2.5 Mealy Machines

Mealy machines are a specific form of state machines (or automata), which are a fundamental concept in computer science. Similar to LTS, Mealy machines provide a formal notation for systems’ behaviors. The main difference is, that a Mealy machine provides an output for any input, which makes them an adequate representation for real-world cyber-physical systems. The definition of Mealy machines reads $M = (Q, \Sigma, \Omega, \delta, \lambda, q_0)$, with $Q$ being the set of states, $\Sigma$ the input alphabet, $\Omega$ the output alphabet (that may or may not identical to the input alphabet), $\delta$ the transition function ($\delta : Q \times \Sigma \rightarrow Q$), $\lambda$ the output function ($\lambda : Q \times \Sigma \rightarrow \Omega$), and $q_0$ the initial state [21]. The transition and output functions might be merged ($Q \times \Sigma \rightarrow Q \times \Omega$). This thesis uses Mealy machines to represent learnt system behavior through observation of inputs and outputs in automata learning (see Section 4.2).

2.6 Automata Learning

Active automata learning is a method of actively querying systems and noting the output to a given respective input. This allows for inferring behavioral models of black-box systems. The classic method of automata learning, called the $L^*$ algorithm, uses the concept of the minimally adequate Teacher [22]. This teacher has (theoretically) perfect knowledge of system to learn and can answer two kinds of questions:

- **Membership queries** and
- **Equivalence queries**.

The membership queries’ answers are denoted in an observation table, that eventually allows for trying to infer an automaton. The equivalence queries determine if the inferred automaton is correct. Lacking a teacher with perfect
system knowledge in a black-box situation, the equivalence queries are ordinarily replaced by traditional conformance testing. More modern algorithms (like TTT [23]) rely on discrimination trees instead of observation tables to be more efficient [24]. This thesis uses both traditional and modern types of automata learning to infer behavioral component models (see Section 4.2).

2.7 Behavioral Equivalences

LTS and automata (particularly of the Mealy type used in this thesis) can be compared for their equivalence. In particular for the purpose of this thesis, an equivalent behavior is important. That means that two automata do not necessarily have to be identical (i.e. all states and transitions being identical), but merely the same input has to yield the same output. This equivalence can be evaluated by trace equivalence (i.e., assessing the same output from the same input) or various degrees of bisimulation [2]. For Mealy machines, bisimulation can be defined (with $Q_1$ and $Q_2$ being two compared Mealy machines as defined in Section 2.5) as [2]:

A) $q_{01} \in Q_1, q_{02} \in Q_2 \cdot (q_{01}, q_{02}) \in R$.

B) for all $q_1 \in Q_1, q_2 \in Q_2 \cdot (q_1, q_2) \in R$ must hold

1) $\sigma \in \Sigma \cdot \lambda_1(q_1, \sigma) = \lambda_2(q_2, \sigma)$

2) if $q_1t \in Post(q_1)$ then there exists $q_2t \in Post(q_2)$ with $(q_1t, q_2t) \in R$

3) if $q_2t \in Post(q_2)$ then there exists $q_1t \in Post(q_1)$ with $(q_1t, q_2t) \in R$

This thesis uses behavioral equivalence for compliance checking (see Section 4.3).
Chapter 3

Research Overview

To increase the comprehensiveness and efficiency of both black box and white box verifying the correctness and security of systems, formalized methods are needed to improve structure and reproducibility. The overall objective is to research more comprehensive and efficient methods for verification through testing. This thesis therefore proposes a structured and automated way to model-based testing in order to leverage this objective at a architectural and at a component level. Although principally applicable to general correctness verification, the thesis ultimately proposes methods for assuring the cybersecurity of and enhancing the trust in systems with a special emphasis on communication protocols used in vehicular systems.

Generally, in the automotive domain, security engineering starts with defining security goals and requirements using a threat modeling process at an architectural level during the design phase. Once the design is implemented, the fulfillment of the derived security requirements has to be verified by testing [17]. As these components are often delivered as black boxes, their verification should not rely on internal systems knowledge. Because of this, the threat model provides context and prioritization of component tests. These tests should check a correct behavior of the system with regard to the security requirements. In other words, the components should comply to a certain specification – this specification is often based on a standard. On the other hand, a threat model is based on certain assumptions about the modeled components during design. If a component implementation later does not comply with the given specification, these assumptions may not hold, which makes the threat model and the resulting test prioritization inaccurate. The overall objective of
Chapter 3. Research Overview

this thesis is to provide a set of formal methods facilitates the automation of test generation from threat models, as well as automatically checking implementations for specification compliance (which again requires an automatic method to derive behavioral component models). As the component behavior has to work black box, because of the reasons stated above, the model generation concentrates on outside interfaces of that component, which is generally a specific implementation of a (standardized or proprietary) communication protocol. Based on our literature survey of previous approaches in these areas, we decided to concentrate on Automata Learning [24] to infer Mealy-type state machines of the behavior of implementations of communication protocols. Then we used trace and bisimulation equivalence checking [25] for compliance checks.

### Overall Objective

Automatically generating formal attack descriptions from architectural models and use automata learning to verify whether the implementation satisfies a standard specification.

#### 3.1 Research Goals

The issues stated above lead to three research goals, namely create working methods to perform:

**RG1** Transforming a system’s threat model into formal descriptions of cyber-attacks.

**RG2** Automatically obtaining state machines of communication protocols from black box scenarios that can be used for correctness and security analysis.

**RG3** Facilitating behavioral equivalence as a method for compliance checking of a learned implementation to a given specification (e.g., a standard).

The attack descriptions (RG1) provide a test scenario in the form of an abstract attack description for the overall system. This scenario is derived from threat modeling the architectural design using a rule set, that scrutinizes potential threats based on the architecture. The state models are inferred from implemented components (RG2) and the compliance checking of these state models (RG3) provide a verdict that verifies compliance to a specification (e.g., a standard). This compliance (RG3) is assumed beforehand in the design phase.
3.1 Research Goals

Figure 3.1: Positioning of the research goals in a structured testing process. Amber denotes artifacts, blue denotes activities, and cyan denotes specification inputs. The arrows denote inputs and outputs, with the dashed input denotes a process including output. The research goals are marked with the dashed red boxes.

during the architectural threat modeling (RG1 - that lead to the attack descriptions). A counterexample regarding the compliance provides (currently manually) input to the rule set for the threat modeling and therefore potentially leads to a different outcome of the threat modeling process. RG3 therefore provides an iterative feedback loop from the implementation back to the design phase and the threat model-based test generation in RG1. This also indirectly brings the standards specification into the threat modeling rule set. Figure 3.1 provides an overview of the research goals in the context of an exemplary automotive cybersecurity testing process. In this figure, amber denotes artifacts, blue denotes activities, and cyan denotes specification inputs. The arrows denote inputs and
outputs, with the dashed input denotes a process including output. The research goals are marked with the dashed red boxes.

3.1.1 Threat Model-based Test Generation

A prominent example of model-based security analysis is the threat analysis and risk assessment (TARA) process widely used in the automotive industry [9]. It uses a threat modeling, based on an architectural design model, to identify threats and prioritize them in order to derive security goals and requirements, which ultimately results in security measures to be implemented in the architecture and components. Some kind of assessment in the fashion of a TARA (although not necessarily the exact same) is even prescribed by the automotive admission process in the UNECE region and the only recognized international standard for implementing a cybersecurity management system [7, 17]. Both admission and standard also mandate to verify cybersecurity measures by testing. In order to create these tests in an efficient manner, another goal of this thesis is to automatically derive test cases from the models made in the design phase to use it later after implementation to verify the efficacy of the planned security measures. The TARA process also determines the verification and validation planning and methods. In this process we included learning-based component testing as presented in RG2 and RG3 [1]. On the other hand, during threat modeling certain assumptions about the model elements are made (e.g., it is assumed that a component’s communication complies with a certain standard) [26, 8]. If these assumptions do not hold, the model becomes inaccurate. It is therefore beneficial for the model’s accuracy that the component’s behavior is checked against the assumptions. When these assumptions can be formulated into a specification, the respective component’s behavior can be automatically checked to comply with that specification. This behavioral compliance checking is formulated in RG3.

**Contributing papers:** Paper I, Paper II, Paper IV.

**RG1:** Create a method to derive formal descriptions of cyberattacks from a system’s threat model.

3.1.2 Automated State Machine Derivation

Formal models have been used very broadly in both research and industrial applications. It is, however, very tedious and costly to create suitable models for correctness and security analysis manually. Furthermore, in some industries
like the automotive, the necessary information to manually creating models might not be present due to very long supply chains and/or non-disclosure. It is therefore beneficial to possess a method to automatically infer formal behavioral models (i.e., state machines) of systems under test in order to foster more efficient analysis and verification processes. As these state machines have to be derived from black box systems (due to the reasons stated above), the interfaces to interact with these systems are their respective implementations of communications protocols. These implementations are the first entry point for adversaries through faults and vulnerabilities. Inferring state machines for correctness and security analysis (as well as test generation) is therefore a significant building block for security improvement. In the course of this, it should also be examined how effective Automata Learning is to infer state machines of industrial real-world communication systems. In an automated process, attack descriptions derived from threat models (RG1) provide the V&V planning for components that should be examined, while the actual checking refers to RG3.

**Contributing papers:** Paper I, Paper II, Paper III.

**RG2:** Create a method to automatically obtain state machines of communication protocols from industrial black box scenarios to use these state machines for correctness and security analysis.

### 3.1.3 Compliance Checking

In many contemporary industries one of the main means for collaborations along the supply chain is written specifications and standards. These include (semi-)formal specifications like development interface agreements (DIAs), specifications in calls for tenders, as well as international and (de facto-) industry standards. To deliver a correct system it is crucial to comply with the respective specification. Furthermore, deviations from standards are most often faults that might lead to security vulnerabilities. Therefore, one goal of this thesis is to create a means for compliance-checking real-world systems in an automated manner. The compliance checking is targeted to be based on state machine models (as derived in RG2), as checking an accurate state machine uncovers consistent and inconsistent behavior both more comprehensively and efficiently, and, thus, more solid than using traditional conformance checking. The specification is modeled into a state machine and its behavior compared to that of learned state machine. Thereby, the behavioral aspect of the equiv-
alence is crucial: it is not necessary that a system’s state machine is *identical* to a specification state machine, only that both state machines *behave* exactly the same. These checks also provide confirmation or rebuttal of assumptions made in threat modeling (based on specifications tailored to these assumptions) creating a link to RG1.

**Contributing papers:** Paper II, Paper III.

| RG3: Demonstrate the applicability of behavioral equivalence as a method for compliance checking of a learned implementation against a given specification (e.g., based on a communication protocol standard) |

### 3.2 Research Method

This thesis follows the Design Science Research Methodology (DSRM) [27]. First thoughts on Design Science were made by Simon in 1969 [28], where he asked how to scientifically scrutinize artificial artifacts of a certain complexity. Artificial in that sense means everything not being deducted strictly by (apodictic) natural laws, i.e., everything human-made, including engineering. Based on this fundament, Nunamaker et al. provided a framework for DSRM for Information Systems (IS) [29]. The framework provides a multi-methodological approach to IS research considering theory building, systems development, experimentation, observation, and their relations to each other. Their work also provides a process to research IS consisting of the following activities and underlying research issues. [29]:

1. **Construct a conceptual framework**
   - State a meaningful research question
   - Investigate the system functionalities and requirements
   - Understand the system building processes/procedures
   - Study relevant disciplines for new approaches and ideas

2. **Develop a system architecture**
   - Develop a unique architecture design for extensibility, modularity, etc.
   - Define functionalities or system components and interrelationships among them
3. **Research Method**

3. Analyze and design the system
   - Design the database/knowledge base schema and processes to carry out system functions
   - Develop alternative solutions and choose one solution

4. Build the system
   - Learn about the concepts, framework, and design through the system building process
   - Gain insight about the problems and the complexity or the system

5. Experiment, observe, and evaluate the system
   - Observe the use or the system by case studies and field studies
   - Evaluate the system by laboratory experiments or field experiments
   - Develop new theories/models based on the observation and experimentation of the system’s usage
   - Consolidate experiences learned

The conceptual framework (1) was done prior to the actual research by defining the overall objective and the research goals (Section 3.1) out of the motivational identified practical problems (Section 1.1), along with studying related work (Section 5). The system architecture (2) was defined in four ways: a) as a structured overall architecture for implementing a testing process (in paper I), b) a system architecture for model learning and model-based compliance testing (papers II and III), c) a conceptual architecture for test case generation from threat models and d) an meta-architecture concept for integrating the components a-c (in this thesis). As system for RG1 was designed (paper IV). Various approaches were considered and examined for RGs 2 and 3 (mainly outlined in papers I, II and III). Based on this evaluation, a system implementation was built for learning and compliance checking, which was also used for extensive system evaluation (paper II).
Chapter 4
Research Contributions

This chapter contains the research contributions that have been made towards reaching the research goals stated in Section 3.1 and outlines the solutions, their novelty and their distribution among the publications. Table 4.1 gives an overview of the contributions of individual papers (outlined in Chapter 4.4) towards reaching the research goals. For each research a different solution is presented, namely:

1. Threat Model-based Test Generation (achieving RG1)
2. Automated Model Derivation and Algorithm Evaluation (achieving RG2)
3. Compliance Checking (achieving RG3)

The threat model-based test generation (1) goes a little bit beyond RG1, as it also partially provides a V&V method selection, although the latter also is meant to contain model checking for test generation, which is reserved for future work (see Section 6.2). The automated behavior model (state machine)
derivation (2) and the compliance checking (3) achieve to RG2 and RG3, respectively. The compliance checking yields a verdict that not only highlights deviations from a specification but also a counterexample that (manually) feeds back into threat model-based test generation by providing input for altering the threat modeling rule set (see Section 3.1). Figure 4.1 provides an overview of the contributions in relation to the research goals and to the thesis papers. Blue boxes mark the contributions, surrounded by the research goals in red dashed lines. The cyan boxes mark previous work the contributions build on, while the dashed black boxes denote the respective papers including the contributions and previous work. The arrows indicate dependencies; solid ones indicate sub-parts and dashed ones indicate inputs.
4.1 Threat Model-based Test Generation

To fulfill RG1, Paper IV presents a method for transferring threat models, via attack trees and labelled transition systems (LTS), into attack descriptions conceived in a domain-specific language (DSL). The method bases on an existing threat modeling tool [30] and an existing DSL called Agnostic domain-specific Language for the Implementation of Attacks (ALIA) [31] and concentrates on the transition between those two. ALIA is a procedural text-based language consisting of sequences of single actions (called test patterns) as a pseudo code that stand for specific steps of a composed attack in a technology-agnostic manner (not bound to a specific system-under-test). These steps will be translated into concrete executable instructions for specific systems-under-test based on Xtext and Xtend [32]. Alia also supports pre and post conditions and flow controls like conditionals and loops. With the labelling function of an LTS, the alphabet of ALIA will be attributed to paths within attack trees generated out of the threat model. Subsequently traversing the resulting LTS along a tree’s path will automatically sequence that input and generate an attack description in ALIA. From the ALIA, it is possible to generate concrete test cases, once an implementation of the architecture is ready. Paper I describes a structured approach to derive a testing strategy from threat modeling RG1 and its connection to learning-based component testing (RG2), and Paper II describes the general embedding of threat modeling into a security testing process.

The approach of transferring a threat model into test descriptions using attack trees and LTS is novel.

4.2 Automated State Machine Derivation

A solution to achieve RG2 was developed by using active automata learning to infer behavioral communication protocol models (concretely state machines, also called automata) on the example of the handshake protocol (ISO 14443-3 [16]) of NFC systems. A similar system for a platooning protocol is in development (see Section 6.2). The developed solution consists of an NFC adapter interface library for the LearnLib library [33], containing the necessary adjustments (including compiling a new firmware) to an NFC hardware adapter along with an abstraction layer that transforms symbols from the learning algorithm to NFC hardware signals and vice versa. The solution also delivers insights on learning the ISO 14443-3 protocol, as it has some very characteristic features in the handshake protocol (particularly, two intertwined combination lock
structures with almost identical states and the property that it does not send a response in case of a non-expected signal). In this setting also, different propagated learning algorithms were evaluated for their suitability and (surprisingly) an older algorithm was found best performing to learn correct implementations (namely the L* algorithm with the closing strategy by Rivest/Schapire), while (less surprisingly) the modern TTT algorithm was best performing when it comes to detecting flaws (see next chapter). The theory for the solution was worked out in Paper I, while Paper II describes the solution concept. Paper III contains the full solution implementation with a description of the complete details for deriving a state machine of NFC handshake protocol implementations.

Specifically, there is quite some previous work on automata learning-based approaches for learning communications systems (even one for NFC banking cards - see Section 5), but for the ISO 14443-3 protocol, so far no automated black box-learning solution has been presented.

4.3 Compliance Checking

RG3 was reached by comparing two automata using bisimulation and trace equivalence: one inferred using automata learning (see 4.2) and a second one based on a specification. While the basic concept was mentioned in Paper II, this was implemented for the NFC handshake protocol in Paper III. This way of checking the conformance is more comprehensive than traditional conformance checking through testing, because it compares the complete behavior of a system with the complete behavior of a specification instead of merely testing a small subset in form of specific traces. The reason for using bisimulation and trace equivalence instead of a much simpler full (graph) identity is that standards compliance does not require the state machines to be identical, but merely to behave the same way. A system with a deviating automaton could still behave equivalent to and therefore be compliant to a specification (or standard).

There is (though few) previous work using the concept of automata learning paired with bisimulation for behavior comparison (see Section 5), however, no solution for practically working protocol performance checking.
4.4 Publications

This section contains an outline of each publication in this thesis consisting of an abstract, the work in this thesis’ context, its contribution to the research goals and the author’s contribution to the respective paper.

4.4.1 Paper I

Title: A Systematic Approach to Automotive Security
Authors: Masoud Ebrahimi, Stefan Marksteiner, Dejan Nićković, Roderick Bloem, David Schögler, Philipp Eisner, Samuel Sprung, Thomas Schober, Sebastian Chlup, Christoph Schmittner, and Sandra König
Abstract: We propose a holistic methodology for designing automotive systems that consider security a central concern at every design stage. During the concept design, we model the system architecture and define the security attributes of its components. We perform threat analysis on the system model to identify structural security issues. From that analysis, we derive attack trees that define recipes describing steps to successfully attack the system’s assets and propose threat prevention measures. The attack tree allows us to derive a verification and validation (V&V) plan, which prioritizes the testing effort. In particular, we advocate using learning for testing approaches for the black-box components. It consists of inferring a finite state model of the black-box component from its execution traces. This model can then be used to generate new relevant tests, model check it against requirements, and compare two different implementations of the same protocol. We illustrate the methodology with an automotive infotainment system example. Using the advocated approach, we could also document unexpected and potentially critical behavior in our example systems.

Work in the thesis context: The paper outlines a structured process to verification by testing, containing threat modeling an black box-inferring behavioral models of systems using automata learning.

Contributes to research goals: RG1, RG2.

Thesis author’s contribution: One equally contributing main author. Main responsible for section 4 (security testing), contributed parts of sections 1 and 2, complete section 4.1 and parts of 4.2. This corresponds to co-developing the overall testing concept based on learning methods, describing the automata learning theory and general parts of the use cases.
4.4.2 Paper II

Title: Approaches for Automating Cybersecurity Testing of Connected Vehicles

Authors: Stefan Marksteiner, Peter Priller, and Markus Wolf

Abstract: Vehicles are on the verge building highly networked and interconnected systems with each other. This requires open architectures with standardized interfaces. These interfaces provide huge surfaces for potential threats from cyber attacks. Regulators therefore demand to mitigate these risks using structured security engineering processes. Testing the effectiveness of this measures, on the other hand, is less standardized. To fill this gap, this book presents a method and a practical implementation that complements traditional conformance testing. We infer a Mealy state machine of the system-under-test.

Work in the thesis context: The paper outlines a concept to automate automotive cybersecurity testing using automata learning, incorporating the results of a threat model, an a platform for automated execution based on a domain-specific language.

Contributes to research goals: RG1, RG2, RG3.

Thesis author’s contribution: Main driver and main author of this paper. Contributed all content except the introductory sections 1 and 2, 3.1, 4.4 and 4.5 (delivered review for these sections). This corresponds with the main concept, an automotive life cycle testing description, a testing process and a model-based testing concept based on automata learning.

4.4.3 Paper III

Title: Using Automata Learning for Compliance Evaluation of Communication Protocols on an NFC Handshake Example

Authors: Stefan Marksteiner, Marjan Sirjani, and Mikael Sjödin

Abstract: Near-Field Communication (NFC) is a widely proliferated standard for embedded low-power devices in very close proximity. In order to ensure a correct system, it has to comply to the ISO/IEC 14443 standard. This paper concentrates on the low-level part of the protocol (ISO/IEC 14443-3) and presents a method and a practical implementation that complements traditional conformance testing. We infer a Mealy state machine of the system-under-test.
using active automata learning. This automaton is checked for bisimulation with a specification automaton modelled after the standard, which provides a strong verdict of conformance or non-conformance. As a by-product, we share some observations of the performance of different learning algorithms and calibrations in the specific setting of ISO/IEC 14443-3, which is the difficulty to learn automata of system that a) consist of two very similar structures and b) very frequently give no answer (i.e. a timeout as an output).

**Work in the thesis context:** This paper contains an examination how to efficiently infer automata of black box NFC systems using automata learning and automatically comparing the behavior (using bisimulation) to a specification automaton, therefore comprehensively assessing the standards compliance of the system-under-test.

**Contributes to research goals:** RG2, RG3.

**Thesis author’s contribution:** Main driver and main author of this paper. Contributed all of the content.

### 4.4.4 Paper IV

**Title:** From TARA to Test: Automated Automotive Cybersecurity Test Generation Out of Threat Modeling  
**Authors:** Stefan Marksteiner, Christoph Schmittner, Korbinian Christl, Dejan Ničković, Mikael Sjödin, and Marjan Sirjani  
**Abstract:** The UNECE demands the management of cyber security risks in vehicle design and that the effectiveness of these measures is verified by testing. This mandates the introduction of industrial-grade cybersecurity testing in automotive development processes. The regulation demands also to keep the risk management current, which again creates the need of stretching the testing over the full life cycle of an automotive system. Currently, the automotive cybersecurity testing procedures are not specified or automated enough to be able to deliver tests in the amount and thoroughness needed to keep up with that regulation, let alone doing so in a cost-efficient manner. This paper introduces an automotive security life cycle governance approach, that takes the currently being developed concepts of Cybersecurity Assurance Levels and Targeted Attack Feasibility into account and provides a means to automatically generate test cases at early development stages. These tests can also be used in later phases to verify and validate the implementations of developed systems. These formalized concepts increase the both the completeness and efficiency of auto-
motivate cybersecurity testing over vehicles’ complete life cycles.

Work in the thesis context: The paper contains an approach to derive an attack tree from a threat model and a concept to transform into an agnostic attack script in a domain-specific language, with a labelled transition system (LTS) as an intermediate step, allowing for automatically creating a test case once the modeled system is implemented.

Contributes to research goals: RG1.

Thesis author’s contribution: Main driver and main author of this paper. Contributed sections 1, 2, 3.2, and 5. This corresponds to the paper’s motivation and a concept for transforming a threat model-based attack tree into attack descriptions written in an (formal) domain-specific language.

4.4.5 Related Papers Not Included in the Thesis

This section outlines papers not directly attributed to the thesis, but demonstrating professional and scientific skills expected for obtaining a licentiate degree. It contains two journal publications [5, 6].

Paper 1

Title: Wireless Security in Vehicular Ad Hoc Networks: A Survey

Authors: Thomas Blazek, Fjolla Ademaj, Stefan Marksteiner, Peter Priller, Hans-Peter Bernhard

Abstract: Vehicular communications face unique security issues in wireless communications. While new vehicles are equipped with a large set of communication technologies, product life cycles are long and software updates are not widespread. The result is a host of outdated and unpatched technologies being used on the street. This has especially severe security impacts because autonomous vehicles are pushing into the market, which will rely, at least partly, on the integrity of the provided information. We provide an overview of the currently deployed communication systems and their security weaknesses and features to collect and compare widely used security mechanisms. In this survey, we focus on technologies that work in an ad hoc manner. This includes Long-Term Evolution mode 4 (LTE-PC5), Wireless Access in Vehicular Environments (WAVE), Intelligent Transportation Systems at 5 Gigahertz (ITS-G5), and Bluetooth. First, we detail the underlying protocols and their architectural components. Then, we list security designs and concepts, as well as the currently known security flaws and exploits. Our overview shows the in-
individual strengths and weaknesses of each protocol. This provides a path to interfacing separate protocols while being mindful of their respective limitations.

**Work contribution in relation the licentiate study:** Demonstrating the ability perform a systematization of knowledge survey of a specialized topic (the security posture of wireless ad-hoc networks relevant to automotive systems).

**Thesis author’s contribution:** Main responsible for the security-related sections of the paper. Selecting relevant protocols in consensus with the co-authors, working out security measures implemented, and security flaws found in the respective protocols.

**Paper 2**

**Title:** A Global Survey of Standardisation and Industry Practices of Automotive Cybersecurity Validation & Verification Testing Processes and Tools  
**Authors:** Andrew Roberts, Stefan Marksteiner, Mujdat Soyturk, Berkay Yaman, Yi Yang  
**Abstract:** The United Nation Economic Commission for Europe (UNECE) Regulation 155 - Cybersecurity and Cybersecurity Management System (UN R155), mandates the development of cybersecurity management systems (CSMS) as part of a vehicle’s life cycle. An inherent component of the CSMS is cybersecurity risk management and assessment. Validation and verification testing is a key activity for measuring the effectiveness of risk management and it is mandated by UN R155 for type approval. Due to the focus of R155 and its suggested implementation guideline, ISO/SAE 21434:2021 - Road Vehicle Cybersecurity Engineering, mainly centering on the alignment of cybersecurity risk management to the vehicle development life cycle, there is a gap in knowledge of proscribed activities for validation and verification testing. This research provides guidance on automotive cybersecurity testing and verification by providing an overview of the state-of-the-art in relevant automotive standards, outlining their transposition into national regulation and the currently used processes and tools in the automotive industry. Through engagement with state-of-the-art literature and workshops and surveys with industry groups, our study found that national regulatory authorities are moving to enshrine UN R155 as part of their vehicle regulations, with differences of implementation based on regulatory culture and pre-existing approaches to vehicle regulation. Validation and verification testing is developing aligned to UN R155 and ISO21434:2021, however, the testing approaches currently used
within industry, utilise elements of traditional enterprise information technology methods for penetration testing and tool sets. Electrical/electronic (E/E) components such as embedded control units (ECUs) are considered the primary testing target, however, connected and autonomous vehicle technologies are increasingly attracting more focus for testing.

**Work contribution in relation the licentiate study:** Demonstrating the ability to systematically conducting a comprehensive literature research (the stance of automotive standards towards cybersecurity testing) and conducting a survey-based research (survey on automotive cybersecurity testing tools and processes used in the industrial practice).

**Thesis author’s contribution:** Main driver and corresponding author of this paper. Contributed sections on global standards and European regional (Germany, France, partly UK) standards, as well as mainly conducting the included industry survey on tools and processes.
Chapter 5

Related Work

This section considers other work in this field and adjacent fields. Given the research goals and contributions it is divided into contributions made in different fields namely: model-based test generation, automated state machine derivation and protocol learning, conformance checking, and applications of automata learning to cybersecurity.

5.1 Model-based Test Case Generation

Model-based testing (MBT) uses a model representation (normally behavioral, but also structural or other kinds) of a system-under-test. Model-based test case generation is an automated test case generation based on model-based testing. It is used in very diverse application domains like Information and Communications Technology (ICT), Automotive, Consumer electronic, Railway, Aerospace, Avionic, Tourism, Agriculture, Finance, Management, Construction, Sport, Automation. The used models include broad variety of different types (like state machines, activity and sequence diagrams, Simulink models, pre/post models, Simulink models, etc.) and the approaches to generate tests include structural coverage (based on control-flow, data-flow, transitions or UML), data coverage (boundary values, statistical or pairwise testing), fault and requirements-based criteria, and explicit and statistical test generation, state, search; model checking, requirements, event, random-based and others [34, 35].
5.2 Attack Trees-based Security Testing

Attack trees are a formal (graphical) representation of the set of possibilities to attack a certain system and have been described in the late 1990ies [10, 11]. They connect specific small attacks (i.e., exploiting threats) to a system in order to attack a complete system or a specific target inside a system with a combined or concatenated attack. The single attacks can be underlaid with different information like necessary skills or features or a success probability, allowing for also calculating this information (i.e., a complete set of skills or features or the combined attack success probability) for the complete attack. It further allows for selecting different paths through that tree that are most efficient regarding defined criteria (e.g., maximized success probability). There is also an approach that combines security-related attack trees with safety-related fault trees and also provides a translation mechanism to transfer them into stochastic timed automata. These can be analyzed using model checking [36]. This can be used to generate test cases. More directly, attack trees have been used to build fault injection-based attacks that can be used directly onto a system-under-test [37]. There is also work to adopt attack trees for automotive systems [38]. There is also an automotive-related method to create attack trees from threat models [26]. The thesis builds upon the later work by providing a translation mechanism from attack trees into a formal attack description language that provides blueprints for cyberattacks in RG1 (Section 3.1.1).

5.3 Formalized Test Descriptions

There is quite extensive work on languages for describing attacks to computer systems [39, 40, 41, 42, 43, 44]. However, this thesis builds upon a domain-specific language (DSL) tailored for automating attacks on automotive systems called Agnostic domain-specific Language for the Implementation of Attacks (ALIA) co-authored by the thesis author [31]. The language concept stems from the principle to abstract attacks on specific automotive systems from their (proprietary) technology-specific traits, leaving a blueprint structure for an attack that needs to be concretized again to be executed against a different system. The ALIA DSL is therefore designed for describing attacks on automotive systems in a technology-agnostic way. Apart from the original intention of porting attacks from one (proprietary) system to another, this allows for specifying attacks at design time and concretizing them once an implementation is available. This thesis integrates this language as a formal description for attacks to
achieve RG1 (Section 3.1.1.

5.4 Automated State Machine Derivation and Protocol Learning

One of the key elements for fully automating model-based test case generation is automatically obtaining a suitable model to analyze. Finite state machines have been frequently used for correctness analyses [45, 46, 47, 48] possibilities to analyze them for their correctness and security properties. There are various approaches to automatically inferring (i.e., learning) state machines. Recurrent networks have been used to learn state machines already in the early 1990ies [49]. Some algorithms work on steering learning from traces by using a two-stage approach. They first analyze traces and mine a rule set and secondly using the rule set for learning automata from traces [50]. Others impose constraints on learning using linear temporal logic [51]. Many of the trace-based inferring methods base on the KTail algorithm [50]. This algorithm has been defined already 1972 by Biermann and Feldman [52]. Trace-based mechanisms are also used to generate other models like sequence diagrams [53]. Since the aim of this thesis is to black box-learning behavioral models of real-world systems, it concentrates on approaches actively querying a system. A method for this that has made many advances in the recent years is automata learning (for the basics see Section 2.6). There is quite some work of using automata learning for security analysis and testing, specifically for learning communication protocols [54, 55, 56, 57, 58, 59, 60]. This also includes NFC but concentrates on the upper layer (ISO/IEC 14443-4) protocol, dodging the specific challenges of the handshake protocol [61]. Aichernig et al. provided a benchmark for different automata learning setups using existing benchmark data [62]. This thesis also provides and automata learning performance evaluation, which is however very specially tailored for the ISO/IEC 14443-3 protocol, with accordingly different results. Recent works also concentrated on making use of these techniques for practical use, e.g., for security analyses [63, 60] or model-based fuzz testing [58]. The thesis differentiates from these works by combining learning with compliance checking and also using this to checking assumptions in threat modeling.
5.5 Conformance Checking using Equivalence of State Machines

There are, partly theoretic, approaches of learning a state machine and comparing it with other ones, targeting target DFAs [64] or probabilistic transition systems (PTS) [65]. For Mealy type machines, which (through their input and output behavior modeling) are better suited for describing reactive systems, Neider et al. provided some fundamental work, using automata learning and bisimulation [66]. Similar things were put into practice by viewing different machines as Labelled Transition Systems (LTS) for model comparison [67, 68] and to verify inferred embedded control software models [69]. However, there is no known comprehensive approach for using bisimulation for protocol compliance checking, which is the differentiation mark of this thesis compared to the described approaches (RG3 - Section 3.1.3)
Chapter 6

Conclusion and Future Work

This chapter summarizes the work included in the Licentiate thesis and outlines further directions to go from the current status of the research done in the thesis and generally in the research field.

6.1 Conclusions

The research described in this thesis aims for facilitating the usage of formal methods for generating tests to assure correctness and security with a focus on the automotive domain and on communication protocols. Following the need of the domain, we concentrated on generating test cases from the security analysis during the design phase, namely Threat Analysis and Risk Assessment (TARA) process and from the implementation, namely by checking the implementation’s compliance with a specification using automata learning. The latter part provides feedback for the design phase: since TARA models systems components with given properties that are based on assumptions about a later implementation (e.g., conforming to international communication protocol standards), the actual compliance to a specification can prove these assumptions to be correct or incorrect. The first part leads to RG1, which is creating attack descriptions out of threat models, which can be used to generate concrete test cases for automotive systems once systems are implemented (Section 3.1.1). The second part is twofold, first we aim for mining a suitable model for security and correctness analysis (RG2 - Section 3.1.2) and second, we aim for a suitable methodology to use a behavioral equivalence with a specification.
as means for compliance checking (RG3 - 3.1.3).

Each of these research goals is met with a respective contribution namely, a method for test generation based on threat models (Section 4.1). This occurs by generating technology-agnostic test specifications written in the Agnostic domain-specific Language for the Implementation of Attacks (ALIA) out of attack trees derived from an existing tool for TARA (ThreatGet) using Labelled Transition Systems (LTS) as a means for the transformation. This approach is, to the best of our knowledge, novel. The second goal is met by automated state machine derivation based on active automata learning (Section 4.2). We showed the practical use of this technique by deriving state machines of Near-Field Communication (NFC) system for correctness and security analyses. We also provide insights on setups, abstraction and performance evaluations of different algorithms in special settings. The third goal was matched by a compliance checking method (Section 4.3). This method compares the behavior of two state machines; one learnt from an implementation and one modeled after a specification (e.g., the ISO 14443-3 standard). We therefore use bisimulation and trace equivalence, which in combination with automata learning is novel for protocol conformance checking.

### 6.2 Future Directions

Despite the efforts taken in the licentiate thesis so far, quite some closely related problems have been left open to fulfill the overall objective in its entirety. Some of these directions are:

- Further implementations of adapters for specific protocols using the same, general learning framework. E.g., dealing with the specifics of V2X protocols or the reader parts of NFC systems in order to create a comprehensive multi-protocol learning framework. The research goal is to create a generally applicable method for protocol model inference.

- Create a method for automated model checking of the learned models for cybersecurity properties. The practical motivation is to automate cybersecurity analysis based on derived models, while the research goal is to derive general rules to check for the security of diverse communications protocols.

- Create a method to derive properties to check from threat models to rigorously check the underlying assumptions.
6.2 Future Directions

Figure 6.1: Positioning of the research goals and future directions in a structured testing process. Amber denotes artifacts, blue denotes activities, and cyan denotes specification inputs. The arrows denote inputs and outputs, with the dashed input denotes a process including output. The research goals are marked with the dashed red boxes.

- Utilize the learned models for fuzz test generation using different strategies based on node and transition properties of the learned models. The research goal is to create highly efficient approaches for fuzz testing in order to create effective zero-input testing methods.

While the first item is basically applying the same principles this thesis uses to new domains, the latter two provide new methods for test case generation. Figure 6.1 shows an overview of the Licentiate research goals in relation with the latter three items above (shapes and colors are the same as in Figure 3.1).
Bibliography


