Thesis for The Degree of Doctor of Philosophy

# On the evaluation of the impact of jamming attacks on cooperative, connected, and automated mobility

Mateen Malik

CHALMERS
UNIVERSITY OF TECHNOLOGY

Department of Computer Science & Engineering
Chalmers University of Technology and Gothenburg University
Gothenburg, Sweden, 2026

**On the evaluation of the impact of jamming attacks on cooperative, connected, and automated mobility**

Mateen Malik

ISBN: 978-91-8103-372-4
DOI: https://doi.org/10.63959/chalmers.dt/5829
Doktorsavhandlingar vid Chalmers Tekniska Högskola
Ny serie nr 5829
ISSN 0346-718X

Department of Computer Science & Engineering
Chalmers University of Technology
SE-412 96 Gothenburg
Sweden
Telephone +46 (0)31-772 1000

This thesis has been prepared using LaTeX.

Printed by Chalmers Digitaltryck
Gothenburg, Sweden 2026.

*To my family*

# Abstract

Cooperative, Connected, and Automated Mobility (CCAM) systems combine connectivity, cooperation, and automation to enable vehicles to exchange information with other vehicles, infrastructure, and remote operators. As these systems depend on wireless communication for safety-critical decision-making, they are inherently vulnerable to communication-jamming attacks. A few examples of these attacks are *communication delay*, *Denial-of-Service* (DoS), *deceptive*, *destructive*, and *barrage jamming*. These attacks have the ability to degrade the availability and integrity of transmitted data. To address this concern, this thesis proposes a reference model for *security benchmarking* that enables structured, repeatable evaluation of CCAM resilience against communication-jamming attacks. Our security benchmarking approach supports both *simulation-based* and *physical* testing. Simulation-based benchmarking is essential for safely conducting large-scale, safety-critical experiments, while physical testing validates simulation outcomes under real-world conditions. The primary components of the benchmarking framework consist of a *driving scenario*, *attack model*, *System Under Test* (SUT), *benchmark execution*, *system response*, and *data analysis*. To support attack modeling, benchmark execution, and outcome analysis, we developed ComFASE, a communication-based fault and attack simulation engine. ComFASE integrates with Plexe, Veins, OMNeT++, and SUMO simulation frameworks. Using ComFASE, we implement and evaluate five jamming attacks injected into the IEEE 802.11p physical-layer model. To showcase the framework's capabilities, we conducted a systematic evaluation and improvement of Cooperative Adaptive Cruise Control (CACC) algorithms. To demonstrate broader applicability, we also evaluate a Remotely Operated Road Vehicle (RORV) application in a Software-in-the-Loop (SiL) environment. Finally, we perform physical barrage-jamming experiments in a Radio Frequency (RF) anechoic chamber to assess the jamming resilience of WiFi communication protocols, the User Datagram Protocol (UDP), and the Secure Reliable Transmission (SRT) protocol used for live video streaming. Overall, this thesis establishes the foundational methods, tools, and analytical insights needed for systematically evaluating CCAM systems through our proposed security benchmarking framework.

**Keywords:** Security Benchmarking, Cooperative and Automated Mobility (CCAM), Cooperative Driving Automation (CDA), Connected Automated Road Vehicles, Cooperative Adaptive Cruise Control (CACC), Network Simulator, Vehicle Simulator, Attack Injection, Jamming Attacks, Barrage Jamming, Wireless Communication, Communication Jamming Attacks, Image Quality Assessment (IQA), Simulation-based Testing, Physical Testing, Teleoperation, Platooning

# List of Publications

## Appended publications

This thesis is based on the following publications:

[A] **Mateen Malik**, Behrooz Sangchoolie, Johan Karlsson, "A Simulation-based Security Benchmarking Approach for Assessing Cooperative Driving Automation (CDA) Applications," In *Proceedings of the 8th EAI International Conference on Intelligent Transport Systems (EAI INTSYS), 2024.*

[B] **Mateen Malik**, Behrooz Sangchoolie, Johan Karlsson, "A Security Benchmarking Approach for Cooperative Driving Automation (CDA) Applications," *Mobile Networks and Applications (MONET), Special Issue on Mobility of Systems, Users, Data and Computing*, 2025.

[C] **Mateen Malik**, Mehdi Maleki, Peter Folkesson, Behrooz Sangchoolie, Johan Karlsson, "ComFASE: A Tool for Evaluating the Effects of V2V Communication Faults and Attacks on Automated Vehicles," In *Proceedings of the 52nd Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)*, 2022.

[D] Mehdi Maleki, **Mateen Malik**, Peter Folkesson, Behrooz Sangchoolie, Johan Karlsson, "Modeling and Evaluating the Effects of Jamming Attacks on Connected Automated Road Vehicles," In *Proceedings of the Pacific Rim International Symposium on Dependable Computing (PRDC)*, 2022.

[E] **Mateen Malik**, Maytheewat Aramrattana, Mehdi Maleki, Peter Folkesson, Behrooz Sangchoolie, Johan Karlsson, "Simulation-based Evaluation of a Remotely Operated Road Vehicle under Transmission Delays and Denial-of-Service Attacks," In *Proceedings of the IEEE 28th Pacific Rim International Symposium on Dependable Computing (PRDC)*, 2023.

[F] **Mateen Malik**, Ludvig Ohlsson, Karthik Sharma, Behrooz Sangchoolie, Johan Karlsson, "Improving the Jamming Resilience of a Cooperative Adaptive Cruise Controller," In *Proceedings of the 9th EAI International Conference on Intelligent Transport Systems (EAI INTSYS)*, 2025.

[G] **Mateen Malik**, Peter Folkesson, Behrooz Sangchoolie, Md Masoom Rabbani, “Evaluating the Impact of Barrage Jamming on the Image Quality of Live Video Streaming,” To appear in *Proceedings of the 21st European Dependable Computing Conference (EDCC)*, 2026.

## Other publications

The following publication is not included in the main licentiate thesis.

[a] **Mateen Malik**, Fredrik Warg, Behrooz Sangchoolie, Md Masoom Rabbani, “Start-Time Sensitivity of Barrage Jamming on Connected Automated Vehicles in Dynamic Scenarios,”
*Under submission.*

# Acknowledgment

First and foremost, I would like to express my sincere gratitude to my main supervisors, Johan Karlsson and Md Masoom Rabbani from Chalmers University of Technology, and Behrooz Sangchoolie, my industry supervisor at RISE Research Institutes of Sweden. I am deeply grateful for their patience, unwavering support, and steady guidance throughout my doctoral journey. Your dedication and integrity have been a constant source of inspiration, and your teachings and influence will stay with me throughout my professional life. I am also very grateful to my examiner, Ioannis Sourdis from Chalmers University of Technology, whose support and constructive feedback have been invaluable to the quality and direction of this work.

Furthermore, I would like to thank all my co-authors, Peter Folkesson, Mehdi Maleki, Maytheewat Aramrattana, and Fredrik Warg, for their meaningful contributions, technical expertise, and fruitful collaborations, which have significantly strengthened this research. I would also like to express my gratitude to all my colleagues at RISE for their continued support, insightful discussions, and collaborative spirit, which helped me achieve my goal.

I am grateful for the support provided by the VALU3S (Grant Agreement No. 876852), SUNRISE (Grant Agreement No. 01069573), AGRARSENSE (Grant Agreement No. 101095835), SYNERGIES, and PRECISE (Reference No. 2024-01718)) research projects. These projects provided not only essential resources but also valuable opportunities for international collaboration, which proved crucial in advancing this research and disseminating its results.

I would like to express special thanks to my manager at RISE Research Institutes of Sweden, Åsa Olsson, for her unwavering support and trust. During challenging times, her understanding made a significant difference, enabling me to balance research work with professional responsibilities.

I would like to thank my mother, Shamim Akhtar, and my brothers, Mubeen Malik and Qadeer Malik, and their families, for their unwavering belief in me and their unconditional support.

Finally, and most importantly, I express my deepest gratitude to my family. To my wife, Qurat-Ul Ain, thank you for your love, patience, and unwavering support. Your strength and understanding have been my constant source of motivation throughout this journey. I am also deeply thankful to my children, Hashim Malik and Ayzal Malik, for their patience and sacrifices during my periods of intense work. I owe you more than words can express.

# Contents

# List of Figures

# List of Tables

# Chapter 1

# Thesis Summary

## 1.1 Introduction

Over the past few decades, vehicles have transformed into intelligent, cooperative, and interconnected cyber-physical systems [1]. These systems can exchange real-time information with other systems and infrastructure by integrating advanced sensing, computation, and wireless communication technologies. Relevant standards that underpin these communication technologies include IEEE 802.11p [2] and C-V2X [3]. Moreover, these communication technologies form the foundation of Cooperative, Connected, and Automated Mobility (CCAM) systems [4], [5], where vehicle cooperation enables functions such as *platooning* [6], *intersection coordination* [7], and *cooperative lane changes* [8]. Through continuous data exchange, CCAM systems improve traffic flow, situational awareness, and overall transportation safety and efficiency.

While connectivity serves as the cornerstone of modern CCAM systems, it also introduces new vulnerabilities and expands the attack surface [9]. As these systems rely on continuous and reliable data exchange for *perception*, *coordination*, and *control*, any disruption or compromise in the *availability*, *integrity*, and *timeliness* [10] of wireless communication can lead to *unsafe*, *degraded*, or *unpredictable* system behavior. An example of such disruptions is communication jamming attacks [11] that pose severe risks to the functionality and dependability [10] of CCAM systems.

In communication jamming attacks, an adversary intentionally transmits interfering radio signals to disrupt or block legitimate wireless communication between systems. By introducing excessive noise or interference into the communication channel, jamming reduces the Signal-to-Interference-plus-Noise Ratio (SINR), causing packet loss, increased latency, or a complete Denial-of-Service (DoS). Such attacks can degrade cooperative decision-making, distort perception data, disrupt or disable communication altogether, all of which are essential for coordinated vehicle functionality. Thus, protecting CCAM systems from communication jamming attacks is essential.

Prior studies [12], [13] have emphasized that safeguarding CCAM systems against such attacks is inherently complex and resource-intensive. It requires systematic testing, validation, and continuous improvement to ensure jamming resilience and trustworthiness.

Testing of CCAM systems can be carried out across multiple phases of the system's development lifecycle using a variety of testing approaches. These testing approaches typically include Model-in-the-Loop (MiL), Software-in-the-Loop (SiL), physical, and real-world testing approaches. Physical testing may take place in controlled settings, such as anechoic chambers and closed proving grounds. In contrast, real-world testing environments include public roads, where systems are exposed to dynamic, unpredictable conditions.

While physical and real-world testing approaches are necessary for validating system functionality under realistic conditions, they present several challenges and concerns. Conducting such tests often involves high operational costs, strict safety requirements, and logistical complexities in maintaining stable experimental conditions. Moreover, real-world and physical tests in an open environment that involve communication-jamming add the risk of signal interference with nearby systems and require compliance with regulatory requirements for radio spectrum use. Moreover, these testing approaches are time-consuming and challenging to reproduce, as environmental factors, such as weather, signal reflections, and hardware variability, can influence results and limit repeatability. Given these challenges and concerns of these testing approaches, simulation-based testing has emerged as a practical and efficient approach for evaluating the interplay between safety and cybersecurity [14] in CCAM systems. In this context, safety means ensuring that a system operates without causing unacceptable harm to people, property, or the environment. In contrast, cybersecurity focuses on protecting the system from unauthorized access, manipulation, or disruption of its data and communication [15].

Simulation-based testing enables the execution of many repeatable, controlled experiments, allowing researchers to expose systems to different types of faults and attacks to assess their resilience systematically. Moreover, simulation-based methods are recognized as a means to reduce development time and cost of CCAM systems [16] while maintaining high testing standards. Regulatory bodies such as the United Nations Economic Commission for Europe (UNECE), which develops internationally harmonized safety and cybersecurity regulations and major automotive OEMs [17], [18] increasingly acknowledge simulation-based testing [19], [20], [21] as an essential component of the verification and validation process for CCAM systems.

While simulation-based testing enables repeatable experimentation, the accuracy of its results depends heavily on the fidelity of the simulation environment. Therefore, physical testing remains essential to validate these results under real-world environmental and operational conditions. These two testing approaches are best seen as complementary: simulation-based testing facilitates rapid iteration and exploration, whereas physical testing provides the ground truth needed to verify system reliability and performance. To systematically compare and assess SUT's functionality against faults and attacks across these environments, a structured evaluation framework is required. Benchmarking provides this framework by enabling the measurement and comparison of system behavior against defined standards or reference scenarios [22]. Crucially, benchmarking can be applied consistently across both simulation-based and physical testing environments, making it an effective tool for evaluating the robustness, safety, and resilience of CCAM systems under varied conditions.

Benchmarking itself has a long-standing role in evaluating system properties across domains such as computing performance [23], transaction processing [24], [25], dependability [22], and security [26]. In the context of CCAM systems, several studies have examined their resilience against cybersecurity threats, particularly against platooning [12], [13], [27], [28], [29]. However, only a limited number specifically address jamming attacks [30], [31], [32]. To the best of our knowledge, no prior study has proposed a framework for defining security benchmarks to evaluate CCAM systems.

In this thesis, we propose a security benchmarking framework for systematically testing, verifying, and evaluating CCAM systems. The framework supports both simulation-based and physical testing methodologies and is designed to assess system resilience against communication-based cybersecurity attacks. To demonstrate its practical applicability, we model and execute several forms of communication-jamming attacks on multiple CCAM-related systems, including vehicle platooning, teleoperation, and wireless video-based safety functions used in automated driving.

To conduct these evaluations, we employ a variety of driving and test scenarios, enabling a comprehensive assessment of system behavior under differing operational conditions. As part of this work, we develop ComFASE, a simulation-based tool that supports benchmark execution, fault and attack injection, and system response analysis. We also construct a physical testbed that enables controlled real-world jamming experiments, which can complement and validate simulation results. Section 1.1.1 outlines our research methodology, presenting the structure and components of the proposed benchmarking framework. Building on this foundation, the thesis addresses the following *overarching research question* (ORQ):

**ORQ** How can a security benchmarking framework be used to systematically assess, compare, and improve the resilience of CCAM systems against communication-jamming attacks across varying attack parameters, scenarios, communication protocols, and testing environments?

## 1.1.1 Research Methology

This section outlines the research methodology adopted in the thesis. Here, we present the primary components of the proposed security benchmarking framework, which structure and connect the contributions of this thesis.

### 1.1.1.1 Security Benchmarking Framework

In our framework, a benchmark is a structured sequence of tests that exposes the system under test (SUT) to specific cybersecurity attacks while recording, monitoring, and analyzing its responses. The primary components of this benchmarking framework include the *attack model*, *SUT*, and *driving scenario*, which together define the stimuli or workload for *benchmark execution*. The output of the *benchmark execution* comprises two primary components: the *system response*, which serves as the basis for *data analysis*.

**Attack Model:** To demonstrate the operation of our proposed security benchmarking framework, we model communication jamming attacks at the physical layer of the wireless communication system. In reference to the jamming taxonomy proposed by Lichtman et al. [11], in a communication jamming attack, an adversary transmits interfering radio signals to disrupt legitimate communication, effectively causing a legitimate communication disruption. As Lichtman et al. emphasize, communication jamming attacks often pursue objectives beyond simple disruption, such as causing unsafe vehicle behavior or collisions. Communication jamming attacks are particularly of interest because they are relatively easy to execute. This type of attack requires limited knowledge of the target system, as attackers can exploit publicly available communication standards to obtain the necessary information to design and launch an attack.

We model five distinct jamming attacks [1]: *delay attack*, *DoS attack*, *deceptive jamming*, *barrage jamming*, and *destructive interference*. These represent varying levels of adversarial capability; in Section 1.1.1.3, we discuss how we modeled these attacks. Each attack model is defined by three primary parameters: *attack value*, *attack start-time*, and *attack duration*. The attack value is tailored to the type of attack being executed. In contrast, the attack start time and duration are tied to the specific test scenario, determining when the attack is launched effectively and how long it persists during the system's operation. These parameters enable flexible, repeatable evaluation of system resilience under varying adversarial conditions.

**System Under Test (SUT):** In the simulation-based testing environment, we evaluated platooning and teleoperation applications. Platooning is a CCAM application in which multiple vehicles drive in close formation, coordinated via Vehicle-to-Vehicle (V2V) communication to improve efficiency and traffic safety. Teleoperation allows a human or machine operator to remotely control a vehicle, typically using real-time video and sensor data over a wireless network. For platooning, we focus on assessing the platoon's string stability and vehicle control under communication-jamming attacks. Specifically, we examined the jamming resilience of four Cooperative Adaptive Cruise Control (CACC) algorithms used for longitudinal vehicle control in a platoon: P1 (Plexe 1) [6], Flatbed [33], Ploeg [34], and Consensus [35]. In addition to platooning, we evaluated the teleoperation application developed by RoboAuto [36]. By incorporating SiL testing, we assess and evaluate the robustness of RoboAuto's software components under jamming conditions.

In the physical testing environment, we expanded our evaluation to include communication protocols commonly used in interconnected CCAM systems. Specifically, we assessed the jamming resilience of User Datagram Protocol (UDP) [37] and Secure Reliable Transport (SRT) [38] protocols by emulating communication jamming attacks that could compromise the integrity and availability of transmitted data. UDP is a lightweight, connectionless protocol that prioritizes low latency over reliability, making it widely used in real-time applications but inherently more vulnerable to packet loss under interference. In contrast, SRT is a connection-oriented protocol designed for secure, reliable, low-latency streaming, featuring packet retransmission, encryption, and congestion control mechanisms that enhance resilience against network disturbances. These protocols are critical for real-time video streaming, which many CCAM systems rely on to carry out safety-critical functions. We evaluated a drone-based surveillance application that provides extended situational awareness and safety support for automated vehicles, particularly in complex or obstructed environments, through live video and sensor data transmission.

[1]In this thesis, the attack model refers to the *impact* of an attack in simulations rather than emulating its actual real-world implementation.

**Driving Scenario:** To evaluate the platooning and teleoperation applications, we employed a range of driving scenarios designed to simulate diverse and challenging operational conditions. These include a braking scenario, where vehicles drive with a constant speed before braking to a complete stop; an acceleration scenario, where vehicles accelerate to a defined acceleration level and then decelerate to a complete stop; and a sinusoidal scenario, where vehicles accelerate and decelerate periodically at a constant frequency to emulate dynamic and extreme driving behaviors. Such driving scenarios are used to test CCAM systems under varying operational conditions [39].

We specifically selected these driving scenarios to expose the systems to both transient and steady-state conditions, which can challenge the control algorithms. The braking and acceleration scenarios help analyze system behavior during abrupt speed changes, while the sinusoidal scenario evaluates the system's ability to maintain synchronization and stability during continuous oscillations. Altogether, these scenarios are used for comprehensive testing to assess the resilience, responsiveness, and safety of CCAM systems under communication jamming attacks. In the physical environment, the test scenario involves communication over a noisy channel, with noise deliberately introduced via communication jamming attacks to evaluate the impact on video quality of the drone-based surveillance system transmitting live video streams to the automated shuttle.

**Benchmark Execution:** For benchmark execution in the simulation-based environment, we utilized ComFASE (Communication-based Fault and Attack Simulation Engine) [40], a tool developed by our research group at RISE and publicly available for download [41]. ComFASE is used to set up attack-injection campaigns, including the modeling and execution of communication jamming attacks. ComFASE is integrated with Veins [42] and three other widely adopted simulation frameworks, Plexe [6], OMNeT++ [43], and SUMO [44], enabling realistic modeling of cooperative adaptive vehicular behavior across various jamming attack scenarios.

For physical test execution, we developed Python scripts to automate control of the jamming device and to manage test suite configuration and execution. The scripts offer flexible control over key attack parameters and enable repeatable, controlled testing of the SUT.

**System Response and Data Analysis:** For data analysis, we use raw data logged from the system's response, including vehicle speed, position, heading, acceleration, deceleration, and collision occurrences. This data forms the basis for classifying outcomes and evaluating overall system performance. Building on this dataset, we refine the analysis according to each testing environment, namely, *simulation-based testing* and *physical testing*.

In *simulation-based testing*, we focus particularly on acceleration and deceleration patterns and collision incidents for analysis. Acceleration and deceleration reveal system behavior, such as comfortable braking versus emergency braking, while collision incidents serve as direct indicators of critical failures triggered by jamming attacks. In *physical testing*, we focus on analyzing the degradation in communication quality caused by jamming attacks, particularly examining how such interference affects data transmitted over different communication protocols.

#### 1.1.1.2 Evaluation Methodology

**Testing Approach:** We use simulation-based and physical testing approaches to evaluate the jamming resilience of CCAM systems. In this context, simulation provides a scalable, repeatable, and safe environment for testing and verification. In contrast, physical testing is essential for validating the accuracy of simulation models and outcomes. Note that the physical testing approach can be used to validate the entire system or a part of it. Simulation-based testing enables both model-in-the-loop (MiL) and software-in-the-loop (SiL) evaluations [45].

In MiL testing, a software model of the system is tested against a simulated model of its environment to verify behavior and accuracy early in the design phase, before any physical hardware exists. SiL testing, on the other hand, evaluates the actual software in a virtual environment, enabling verification of functionality, timing, and robustness before actual deployment.

Fig. 1.1 illustrates the complete chain for testing, verification, and validation (V&V) approaches of CCAM systems, highlighting the transition from purely virtual testing environments to system integration and validation. These testing approaches support early vulnerability identification, ensuring that CCAM systems are robust, resilient to cyberattacks, and capable of operating safely under dynamic driving scenarios.

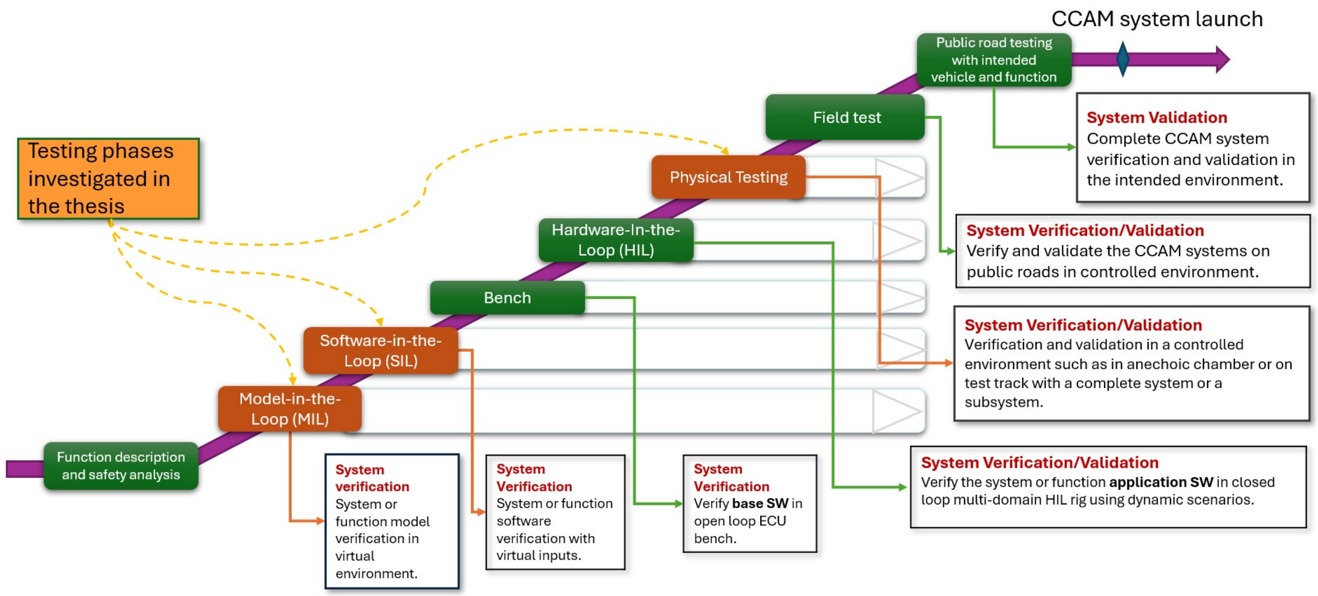
CCAM system launch
Public road testing with intended vehicle and function
Field test
Physical Testing
Hardware-In-the-Loop (HIL)
Bench
Software-in-the-Loop (SIL)
Model-in-the-Loop (MIL)
Function description and safety analysis
Testing phases investigated in the thesis
System Validation
Complete CCAM system verification and validation in the intended environment.
System Verification/Validation
Verify and validate the CCAM systems on public roads in controlled environment.
System Verification/Validation
Verification and validation in a controlled environment such as in anechoic chamber or on test track with a complete system or a subsystem.
System Verification/Validation
Verify the system or function application SW in closed loop multi-domain HIL rig using dynamic scenarios.
System Verification
Verify base SW in open loop ECU bench.
System Verification
System or function software verification with virtual inputs.
System verification
System or function model verification in virtual environment.

Figure 1.1: Complete chain for testing, verification, and validation of CCAM systems, where white arrows indicate that test phases may continue in parallel with subsequent phases.

Table 1.1 provides an overview of our key contributions across the different testing approaches, including the types of jamming attacks performed against which we evaluated the SUT, and associated test scenarios.

**Testing Technique:** *Fault injection* [46], [47], [48] and *attack injection* [49], [50] are two common techniques used to evaluate the resilience of CCAM systems. Other complementary testing methods include scenario-based testing, acceptance testing, performance testing, and various functional and non-functional evaluation testing techniques. These testing techniques are systematic methods for assessing the robustness, safety, and security of cyber-physical systems by deliberately introducing controlled disturbances or adversarial actions.

In fault injection, artificial faults, such as bit flips [51], [52], or sensor malfunctions [53], [54] are introduced to study how the system detects, tolerates, and recovers from faults. From an ISO 26262 [55] perspective, fault injection is a recommended verification activity for demonstrating that safety mechanisms can handle relevant faults and that safety goals are met. It provides evidence that the system behaves safely under realistic fault conditions. In contrast, attack injection emulates malicious behaviors, such as jamming, spoofing, or DoS attacks [50], to assess the system's resilience against cybersecurity attacks.

These testing approaches can be performed in both simulation-based environments and physical testbeds, enabling researchers to analyze system behavior under stressed or compromised conditions without risking permanent damage to the system or the surrounding environment.

We used an attack injection technique where we introduced communication jamming attacks [56] to evaluate the jamming resilience of the target SUTs. In our attack-injection testing technique, malicious wireless communication signals are injected either by altering the physical-layer waveform of a legitimate signal or by injecting interfering signals at the physical layer, with the explicit goal of disrupting communication to assess their impact on the system's safety-critical functionalities.

#### 1.1.1.3 Modeling of Jamming Attacks

This section provides details on the attacks we modeled in the simulations, including the key simulation parameters and how they are used to model the attack. The attacks models include *delay*, *DoS*, *barrage jamming*, *deceptive jamming*, and *destructive interference*.

In the **delay** attack, messages are intercepted and prevented from reaching their intended recipients. These intercepted messages are later retransmitted to the system, causing congestion at the receiver. We model the delay attacks by manipulating the Veins simulation parameter called *propagation delay*. Veins is an open-source simulation framework designed for modeling and evaluating vehicular network communication. The *propagation delay* parameter is originally designed to be used to calculate the propagation delay of the wireless channel, which, in our case, is the 'free space path loss' (FSPL) wireless channel model. Under normal communication conditions, propagation delays are minimal (e.g., a 300-meter transmission distance corresponds to roughly 1

Table 1.1: Overview of our contributions across test approaches, jamming attack types, and tools used, SUT, scenarios, and outcome classification parameters.

| | | **Testing approach** | | |
|---|---|---|---|---|
| | | **MiL** | **SiL** | **Physical** |
| Communication Jamming Attacks | Delay | ✓ | ✓ | |
| | Denial-of-Service (DoS) | ✓ | ✓ | |
| | Deceptive | ✓ | | |
| | Destructive Interference | ✓ | | |
| | Barrage | ✓ | | ✓ |
| System Under Test (SUT) | Platooning | ✓ | | |
| | Teleoperation | | ✓ | |
| | Drone Surveillance | | | ✓ |
| Driving Scenario | Acceleration | ✓ | ✓ | |
| | Deceleration | ✓ | ✓ | |
| | Sinusoidal | ✓ | | |
| Test Scenario | Communication Over Noisy Channel | | | ✓ |
| Attack Injection Tool | ComFASE | ✓ | ✓ | |
| | Python-based Scripts | | | ✓ |
| Outcome Classification Categories | Safe Braking | ✓ | ✓ | |
| | Emergency Braking | ✓ | | |
| | Collision Incidents | ✓ | | |
| | Normal Operation | ✓ | ✓ | |
| | Speed Deviation | | ✓ | |
| | Disconnection | | ✓ | |
| | Image Quality Degradation (Perfect, Degraded, Lost) | | | ✓ |

µs of delay). In contrast, our minimum injected delay is set to 1 second, which is several orders of magnitude larger than the value intended under normal (attack-free) conditions. When these delayed packets are released, they often arrive simultaneously with newly generated messages, resulting in channel congestion and increased queuing.

In contrast to delay attacks, **Denial-of-Service (DoS)** attacks completely block wireless communication channels for an extended period. We model the DoS attacks by manipulating the same *propagation delay* parameter we used to model communication delays. However, in DoS attacks, the delay persists until the end of the simulation, effectively preventing message delivery altogether and simulating the impact of prolonged channel unavailability.

In our approach to modeling the 'delay' and 'DoS' attacks, we focus solely on their timing effects; packet content remains unchanged. To ensure a systematic approximation of these attacks, we uniformly apply the injected delay across all communication links between the transmitting and receiving vehicles. This means every inter-vehicle communication path experiences the same level of delay, allowing us to study the global impact of these attacks on platoon stability. Note that, as the propagation delay mechanism in Veins does not inherently support message interception, buffering, or replay, our approach should be viewed as an engineering approximation. It captures the timing-related impact of delay and DoS attacks, rather than providing an accurate representation of their underlying physical or protocol-level behaviors.

In **barrage jamming** attacks, the adversary transmits high-power noise over a broad spectrum of frequencies and, as a result, wholly or partially blocks the transmission or reception of legitimate signals. We model barrage-jamming attacks by manipulating the Veins simulation parameter, *Noise*. **Deceptive jamming** involves transmitting fake signals that mimic legitimate signals in frequency and power, confusing receivers and consequently causing them to process false or misleading information. We model deceptive jamming attacks by manipulating a Veins parameter, *Interference*.

It is worth mentioning that the *noise* and *interference* parameters in the Veins simulator are designed initially to support the receiver's physical-layer model when computing the SINR of incoming packets. From the receiver's perspective, both increased noise and increased interference reduce the SINR in a similar manner, effectively lowering the likelihood of successful packet reception.

**Destructive interference** occurs when a well-crafted malicious signal in terms of time, frequency, and space is transmitted to interfere with legitimate signals destructively, causing signal distortion or signal cancellation. We model destructive interference by integrating a destructiveness parameter $D$ into the Veins simulator. This parameter multiplies with the legitimate signal power of a given data channel, causing destructive interference.

Note that, in practice, signal cancellation requires the exact phase of the receiving signal, timing, and channel-state knowledge that a single scalar parameter cannot capture. Our attack model, therefore, parameterizes a spectrum of destructive interference, from minor distortion to complete cancellation of the legitimate signal at the receiver. Complete cancellation, also known as *nulling*, represents an idealized worst-case scenario. Achieving nulling in the real world is difficult because the wireless channel is dynamic (time-varying multipath, mobility, and synchronization) and would need perfect 'Channel State Information' (CSI) and phase coherence.

**In summary**, in our simulation setup, we model the *impact* of jamming attacks rather than emulating their exact real-world implementation. For example, instead of emulating the full mechanics of a real delay attack, such as intercepting, buffering, and later retransmitting signals to the receiver, we model only its impact by introducing delays at the physical layer of the communication system. This provides an opportunity to analyze the system-level impact of the attack in a simple, controlled, and repeatable way. In a real-world context, different jamming attacks, such as *barrage*, *deceptive*, and *destructive* jamming, affect the wireless channel in different ways. For example, *deceptive jamming* can generate signals that closely resemble legitimate transmissions, exploiting protocol timing structures, or selectively target specific portions of the frequency spectrum.

Nevertheless, using our simulation setup, we can still approximate the resulting *impact* on communication performance, enabling us to study attack consequences qualitatively similar to those observed in real-world implementations.

While our simulation models capture the *qualitative* impact of communication jamming attacks on the system safety, they should not be interpreted as providing quantitatively precise thresholds for real-world attack injection scenarios. Instead, they offer a controlled, systematic, and repeatable approximation, highlighting the need for more detailed physical-layer studies and higher-fidelity wireless channel models to design representative jamming attack models and mitigation strategies. Therefore, in one of our works, we generated real communication jamming signals and conducted an actual barrage jamming attack, rather than modeling its impact in simulation. To do this, we developed and implemented methods and tools to introduce jamming attacks in a controlled, real-world environment, enabling analysis of system behavior and safety under realistic communication interference.

### 1.1.2 Research Contributions

The primary contributions of this thesis are:

**C1** Proposing a security benchmarking framework for evaluating the resilience of CCAM systems against communication-based cyberattacks. [*Chapter 2 (Paper A), and Chapter 3 (Paper B)*]

**C2** Developing a Communication-based Fault and Attack Simulation Engine (ComFASE), which allows comprehensive attack modeling, configuration, and experimentation. To demonstrate the engine's applicability and capabilities, we modeled two types of jamming attacks: *delay attacks* and *denial-of-service (DoS) attacks.* [*Chapter 4 (Paper C)*]

**C3** Extending ComFASE with three additional representative jamming attack models: *barrage jamming*, *destructive jamming*, and *deceptive jamming.* This enables us to perform an in-depth analysis of the impact of jamming attacks on a platooning system consisting of four vehicles. [*Chapter 5 (Paper D)*]

**C4** Studying the impact of attack model parameters, including *attack start time*, *attack duration*, and *attack value*, on the jamming resilience of the platooning system. [*Chapter 2 (Paper A), Chapter 3 (Paper B), Chapter 4 (Paper C), Chapter 5 (Paper D), Chapter 6 (Paper E)*]

**C5** Analyzing and comparing the impact of barrage jamming attacks on platoon string stability and CACC algorithm functionality under two driving scenarios: *sinusoidal* and *braking.* [*Chapter 2 (Paper A)*]

**C6** Improving the jamming resilience of a Cooperative Adaptive Cruise Control (CACC) algorithm that is implemented based on Constant Vehicle Spacing (CVS) control policy. [*Chapter 7 (Paper F)*]

**C7** Evaluation of a SIL application software of a remotely operated road vehicle under *transmission delays* and *DoS attacks.* [*Chapter 6 (Paper E)*]

Table 1.2: Mapping of research questions (RQ) to the thesis chapters.

| **RQ** | **Reference to Chapters** |
|---|---|
| Q1 | Chapter 2 (Paper A), Chapter 3 (Paper B), Chapter 7 (Paper F) |
| Q2 | Chapter 4 (Paper C), Chapter 5 (Paper D), Chapter 6 (Paper E) |
| Q3 | Chapter 2 (Paper A) – Chapter 7 (Paper F) |
| Q4 | Chapter 2 (Paper A) |
| Q5 | Chapter 2 (Paper A) |
| Q6 | Chapter 7 (Paper F) |
| Q7 | Chapter 8 (Paper G) |

**C8** Developing and evaluating a methodology and a testbed for introducing and analyzing the impact of *barrage jamming* attacks on drone communication in the physical (as opposed to simulation) environment. [*Chapter 8 (Paper G)*]

### 1.1.3 Research Questions

Building on the *Overarching Research Question* (ORQ) defined in Section 1.1, this section introduces a set of detailed and focused research questions. These research questions decompose the ORQ into specific aspects of security benchmarking, the influence of different jamming techniques, the role of attack parameters, the impact of driving scenarios and vehicle behaviour, the resilience of cooperative control algorithms, and the performance of communication protocols. Together, they enable a systematic and comprehensive assessment of CCAM system behaviour under communication-jamming attacks.

Our primary focus is on vehicle platooning, where we introduce multiple jamming attack types and evaluate the jamming resilience of several CACC algorithms. Beyond the platooning application, we extend our framework to a Software-in-the-Loop (SiL) implementation of a Remotely Operated Road Vehicle (RORV). Here, we assess how communication degradation, through delays, message losses, and outages, affects remote control responsiveness and the activation of safety mechanisms. These investigations contribute to answering the research questions related to the consequences of jamming on system behavior in CCAM teleoperation contexts. We run controlled experiments in a physical testbed to evaluate the effects of barrage-jamming on wirelessly transmitted video streams. These experiments support the research question regarding communication protocol resilience.

All together, this thesis demonstrates the applicability of the proposed security benchmarking framework and systematically addresses seven research questions (Q1–Q7), thereby providing a cohesive foundation for answering the overarching research question. Table 1.2 provides a concise mapping of the research questions to the chapters in which they are addressed. The first research question is formulated as follows.

**Q1** How can a security benchmarking framework be designed and applied to systematically identify vulnerabilities, evaluate jamming resilience, and compare the performance of alternative CCAM system solutions?

We address Q1 across Chapters 2, 3, and 7. Chapter 2 (Paper A) evaluates the resilience of a specific CACC algorithm under various jamming attacks. Chapter 3 (Paper B) extends this investigation by comparing the jamming resilience of several CACC algorithms. Chapter 7 (Paper F) builds on these insights by introducing two enhanced, jamming-resilient variants of the CACC algorithm previously analyzed in Chapter 5 (Paper D). These new variants, developed in Chapter 7, demonstrate improved robustness against communication jamming attacks.

**Q2** To what extent does the choice of jamming technique influence the likelihood of a successful jamming attack in terms of emergency braking or vehicle collisions?

We address Q2 in Chapters 4 (Paper C), 5 (Paper D), and 6 (Paper E). Chapter 4 provides a comprehensive analysis of the effects of communication delays and DoS attacks on a system-under-test comprising a platoon of four vehicles operating with a CACC algorithm. Chapter 5 examines the impact of *barrage jamming*, *deceptive jamming*, and *destructive interference* attacks. Chapter 6 analyzes the impact of *delay* and *DoS* attacks on remotely operated road vehicles. In this chapter, we evaluate the SIL components of the teleoperation system provided by RoboAuto [36].

**Q3** How do the parameters of our attack models, i.e., the attack start time, attack duration, and attack value, affect the outcomes of the jamming attacks?

In Q3, we aim to investigate how variations in the attack model parameters influence the distribution of outcomes. We characterize each attack model by three fundamental parameters: the *attack start time*, the *attack duration*, and the *attack value*.

The first two parameters are relative to the time axis of the driving scenario, which specifies the temporal evolution of the vehicle system's motion. The *attack value* determines the intensity or strength of the malicious signal transmitted to the receiver and varies depending on the specific jamming technique. For instance, in a *barrage jamming* attack, the attack value corresponds to the power of the interfering noise signal. In contrast, in a *destructive interference* attack, it represents the degree of destructiveness imposed on the legitimate communication signal.

We address Q3 in Chapters 2–7. In Chapters 2, 3, and 7, we analyze the impact of attack parameters on *barrage jamming* attacks against different CACC algorithms. In Chapter 4, we examine the impact of the attack parameters for *delay* and *DoS* attacks. Chapter 5 examines the impact of the attack parameters for *barrage jamming*, *deceptive jamming*, and *destructive interference* attacks. Similar to Q1, Chapter 6 analyzes the impact of *delay* and *DoS* attack parameters on remotely operated road vehicles.

**Q4** How does the driving scenario influence the outcomes of jamming attacks on a platoon?

The fourth research question addresses the influence of the driving scenario on the outcomes of jamming attacks. To investigate this, we conduct jamming experiments across two representative driving scenarios: the *sinusoidal* and *braking* scenarios. These scenarios are included in the Plexe simulation framework. We address Q4 in Chapter 2.

**Q5** Are certain vehicles in the platoon more likely to cause collisions under jamming attacks?

The research question Q5 focuses on attacks that result in vehicle collisions, specifically on identifying the vehicle that hits another vehicle from behind (a rear-end collision). We refer to this vehicle as the *collider vehicle*. By analyzing the conditions under which such collisions occur and the characteristics of the collider vehicle, we aim to gain deeper insights into how jamming attacks propagate through the platoon. We address Q5 in Chapter 2.

**Q6** To what extent can a CACC algorithm based on a constant vehicle spacing (CVS) policy achieve jamming resilience?

Addressing this question is of interest as CACC algorithms based on Constant Vehicle Spacing (CVS) policy are inherently more vulnerable to jamming attacks due to their heavy reliance on inter-vehicle communication. Unlike CACC algorithms that employ Constant Time Headway (CTH) policy, which relies on local sensor data (e.g., radar or lidar), CVS-based CACC algorithms depend almost entirely on the timely reception of cooperative information. A loss or degradation of this information caused by jamming can therefore lead to instability, unsafe spacing within the platoon, or even collisions.

We investigate whether it is possible to enhance the jamming resilience of CVS-based CACC algorithms to achieve performance comparable to or better than that of CTH-based CACC algorithms, thereby improving safety and reliability in communication-dependent platooning systems. We address this research question in Chapter 7.

**Q7** How does a barrage jamming attack impact the quality of data transmitted over different communication protocols in a physical, real-world environment compared to a simulation setting?

So far, we have executed our attack test campaigns in a simulation environment. Q7 addresses the challenges and opportunities of introducing barrage jamming attacks in a physical environment. To answer Q7, we design and execute a barrage jamming experiment in a physical setup, where we injected attacks using fixed parameters, including the *attack start time*, *attack duration*, *attack value*, *distance*, *geometry* (jammer–receiver separation and antenna orientation), and *duty cycle* (continuous transmission). We then carry out a series of controlled test campaigns using two communication protocols: UDP [37] and SRT [38].

We then evaluate the impact of barrage jamming on wireless communication quality in a controlled real-world environment (physical testing). We analyze how each protocol responded to interference in terms of reliability and transmitted video quality. The analysis focuses on both physical-layer metrics, such as *Peak Signal-to-Noise Ratio (PSNR)* [57]. Application-layer metrics relevant to video data transmission, including *Video Multi-Method Assessment Fusion (VMAF)* [58] and *Multi-Scale Structural Similarity Index Measure (MS-SSIM)* [59]. These metrics collectively provide quantitative insights into how barrage jamming attacks degrade perceived video quality under realistic operating conditions. We address Q7 in Chapter 8 (Paper G).

It is important to note that implementing physical jamming attacks is a necessary step toward validating simulation-based attack models. However, validating the barrage jamming attack modeled in our simulations against a corresponding real-world attack is not included in this work. Such validation requires additional in-depth analysis and is planned as part of our future work to strengthen the simulation's fidelity and applicability further.

### 1.1.4 Thesis Structure

The remainder of this thesis is organized as follows. Section 1.2 provides a concise overview of the background and related work relevant to this research. Section 1.3 summarizes the scientific papers that constitute the main contributions of the thesis. Chapters 2 to 8 contain the reprints of these papers, each representing a distinct part of the overall research contribution.

## 1.2 Background

This section presents the essential background information that underpins our research. Section 1.2.1 introduces the fundamental wireless communication concepts relevant to this study, while Section 1.2.2 summarizes the various communication jamming techniques described in the literature. In Section 1.2.3, we describe the platooning application, the CACC algorithms investigated in our simulation-based attack injection experiments. Finally, we conclude this section with a related research, presented in Section 1.2.4.

## 1.2.1 Wireless Communication

In Section 1.2.1.1, we give a brief overview of the IEEE standards for wireless access in vehicular environments, while elaborating on data transmission and reception at the physical layer, antenna characteristics, and wireless channel behavior in Section 1.2.1.2. Section 1.2.1.3 describes the wireless communication protocols that we used for testing in the simulation and physical environments.

### 1.2.1.1 Wireless Access in Vehicular Environments

The IEEE family of Wireless Access in Vehicular Environments (WAVE) [60] standards defines the protocol stack and services that enable short-range vehicular communications (DSRC/WAVE), spanning application-level services, security, networking, multi-channel operation, and the wireless MAC/PHY. Essential elements of the WAVE suite include the resource manager (IEEE 1609.1), security (IEEE 1609.2), networking (IEEE 1609.3), and multi-channel/MAC operations (IEEE 1609.4). At the same time, IEEE 802.11*p* [2] specifies the lower MAC and physical layers used for vehicular wireless access.

The Veins simulation framework provides dedicated models for the DSRC and WAVE communication stack (including IEEE 802.11*p* and IEEE 1609.4 behaviour), thereby offering a suitable environment for PHY/MAC-level experiments and attack injection in vehicular scenarios. In our experiments, we implement communication jamming attacks by modifying parameters in the IEEE 802.11*p* physical-layer model available in Veins. For communication, we use a message update rate of 10 Hz with each broadcast message sized at 200 bytes. Consequently, a one-second outage results in the loss of 10 messages per sender, for example, in a four-vehicle platoon, this corresponds to a total of 40 messages lost during that outage.

### 1.2.1.2 V2V Physical Layer Communication

**IEEE 802.11p Transceiver:** This section summarizes the transmitter and receiver blocks of a typical IEEE 802.11*p* system [62], as shown in Fig. 1.2. IEEE 802.11*p* operates in the 5.9 GHz frequency band, with a 10 MHz channel bandwidth, and uses orthogonal frequency-division multiplexing (OFDM) to transmit data efficiently. OFDM divides the 10 MHz channel into 64 subcarriers, of which 52 are used: 48 for data subcarriers and 4 for pilot subcarriers. The pilot subcarriers play an essential role in wireless OFDM-based systems. They provide a reference for accurately estimating and compensating for frequency offset and phase noise in the received signal. The remaining 12 subcarriers act as guard bands to prevent interference with adjacent channels.

The forward error correction (FEC) scheme used for IEEE 802.11*p* transmission is a convolutional code with industry-standard generator polynomials $g_o$ = 133 and $g_1 = 171$ with supported code rates of 1/2, 2/3, and 3/4 [62]. FEC is a technique used in digital communication systems to improve the reliability of data transmission over noisy or unreliable channels. The idea behind FEC is that the transmitter adds redundant error-correcting codes to the original data

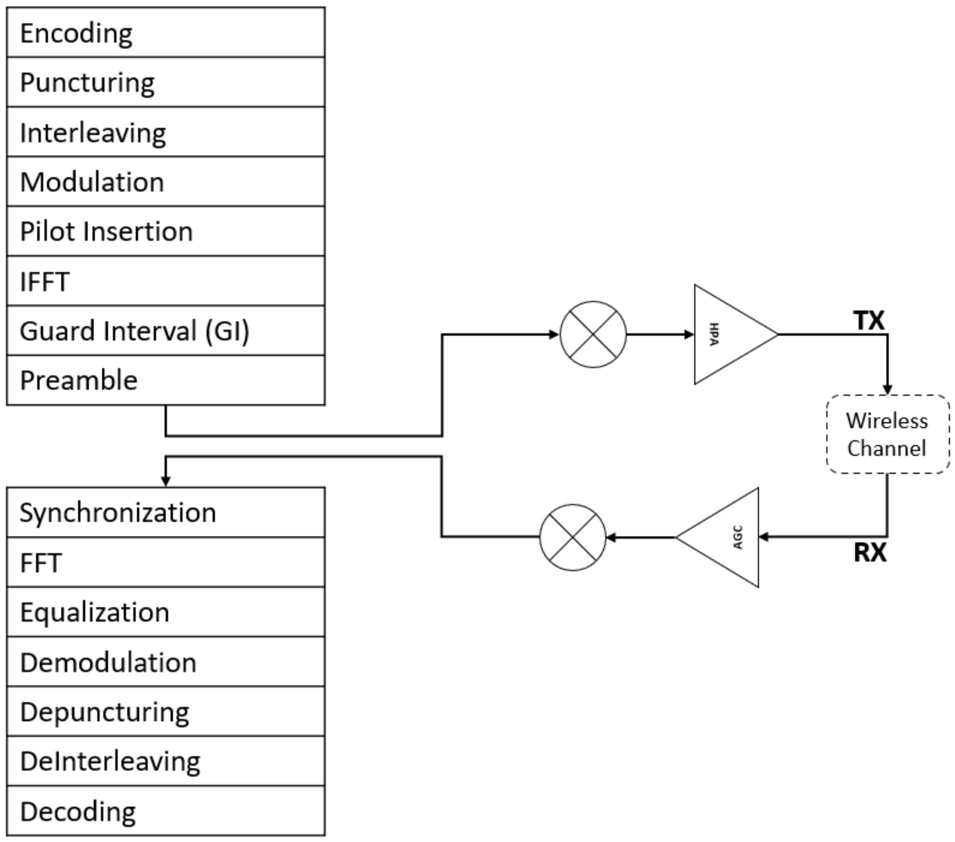

Figure 1.2: A simplified block diagram of IEEE 802.11p physical layer transceiver design [55], [61].

before sending it. These redundant bits allow the receiver to detect and correct errors that occur during transmission without the need for retransmission [63].

Code rates higher than 1/2 are achieved through puncturing [62], a process that selectively removes certain bits from the encoded data stream. By eliminating specific bits, puncturing effectively increases the code rate without changing the structure of the convolutional code [64]. The receiver can recover the original data by knowing which bits were punctured and applying error correction. Interleaving is a process that comes after puncturing and before modulation. It rearranges the bits across the transmission frame, aiming to spread consecutive bits across different subcarriers and time slots. This helps mitigate burst errors that may affect groups of consecutive bits due to interference, fading, or other channel disturbances. The de-interleaving reorders the bits to their original positions at the receiver, reversing the interleaving pattern applied at the transmitter.

For transmission, several modulation schemes are used in the context of V2V communication [65]. These modulation schemes include binary phase shift keying (BPSK), quadrature phase shift keying (QPSK) [66], and quadrature amplitude modulation such as 16-QAM and 64-QAM [62].

After modulation, known reference signals called pilots are added to specific subcarriers for channel estimation. An inverse fast fourier transform (IFFT) is then performed to convert the signal from the frequency domain to the time domain, enabling multi-carrier transmission. Guard interval is used to mitigate inter-symbol interference (ISI), where a portion of the time domain signal is copied from the end and appended to the beginning of the signal.

Before forming the OFDM symbol, a predefined sequence of time domain signals, known as the preamble, is added for synchronization, initial channel estimation (e.g., fading, attenuation, and phase shifts), and receiver initialization to correctly interpret incoming signals. After completing these processing steps, the OFDM symbol is amplified using a high-power amplifier (HPA) before transmission (see Fig. 1.2).

The modulation schemes used in the Veins simulation framework are BPSK and QPSK, depending on the required data rate and the wireless channel conditions. BPSK is more resistant to noise and multipath fading, while QPSK offers a higher data rate than BPSK is less resilient to noise and interference. It has a coding rate of 1/2 and a data rate of 6 Mb/s with a channel spacing of 10 MHz [42].

In OFDM-based communication systems [67], the receiver evaluates whether the received signal has been correctly decoded and matches the originally transmitted data. There is an upper bound for the number of erroneous bits in a packet that a receiver can correct to preserve the integrity of the data. If the number of erroneous bits exceeds that threshold, the packet cannot be decoded correctly, preventing the successful recovery of the transmitted data.

BER is a metric that quantifies the ratio of bits received in error to the total number of bits transmitted. BER is calculated based on the signal interference and noise ratio (SINR) and the given modulation scheme. SINR determines the quality of the signal received. It is the ratio between the legitimate signal power and the total power of noise and interference. Noise is the unwanted channel disturbances, parasitic noise, and interference is the transmission of the neighboring channels using the same frequencies (see Eq. 1.1).

$$SINR = \frac{SignalPower}{InterferencePower + NoisePower} \tag{1.1}$$

In the Veins simulations, there is a module at the physical layer called the ‘decider, which computes the packet’s SINR and inputs it into a bit error model to calculate the BER. This model calculates the bit error rate based on the modulation scheme and SINR. The BER is then compared with a randomly generated number between 0.0 and 1.0. If the generated number exceeds the BER, the packet is considered error-free and passed to the next layer [68]. Veins utilizes BER and a random number generator to simulate the unpredictable nature of real-world errors.

**Wireless Channel:** A wireless channel is the medium through which wireless communication signals are transported from a transmitter to a receiver. It

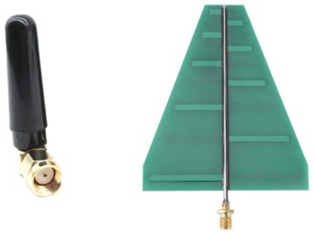

Figure 1.3: An example of a monopole antenna (on the left) and a directional antenna (on the right).

consists of three essential elements: the transmit antenna, the air medium, and the receiver antenna. There are different types of *antennas* used for signal transmission, such as *monopole* and *directional* antennas [69] as shown in Fig. 1.3. The monopole antenna transmits the electromagnetic waves equally in all directions, whereas in the case of a directional antenna, the concentration of electromagnetic waves is in one direction [69] as illustrated in Fig. 1.4. In the

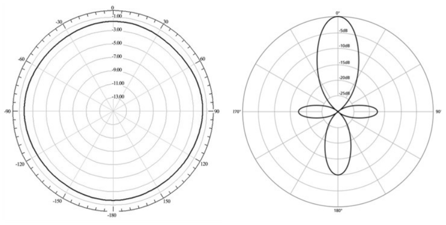

Figure 1.4: Radiation pattern of monopole antenna (on the left) and a directional antenna (on the right).

platooning application model, which is part of our simulation environment, all vehicles are equipped with monopole antennas for transmitting and receiving signals [70]. If the received signal power is above the antenna's sensitivity threshold, the signal is sent to the physical layer for processing [42]. The default threshold value in Veins is -95 $dBm$.

A wireless channel occupies a given set of frequencies within a frequency band in the electromagnetic spectrum. The key factors that affect the signal in the wireless channel include *(i)* the distance *'d'* between transmitter and receiver, *(ii)* the sensitivity of the receiver antenna, *(iii)* the wavelength '$\lambda$' of the transmitted signal, and *(iv)* reflections from objects in the environment such as road, buildings, and trees. The Veins simulator includes three predefined channel models [71] for calculating signal attenuation which is the reduction in signal strength as a signal travels through a wireless channel. These models are *Two-Ray Interference Model*, *Obstacle Shadowing*, and *Free Space Path Loss (FSPL)* models. The Two-Ray Interference Model considers two primary propagation paths from the transmitter to the receiver: direct line-of-sight (LOS) and ground-reflected path. In the obstacle shadowing environment model, the signal attenuation is caused by physical obstacles, such as buildings, trees, or vehicles, that partially or fully obstruct the line of sight between the transmitter and receiver.

The Free Space Path Loss (FSPL) model attenuates the signal as it travels in free space without any obstacles or reflections. FSPL is determined by the distance $d$ between the transmitter and receiver and the signal's wavelength $\lambda$. This model is widely used in wireless communication to estimate signal loss under ideal line-of-sight conditions [68]. The mathematical representation of *FSPL* is called the *Friis transmission equation* shown in Eq. 1.2.

$$Pr[dBm] = Pt[dBm] + Gt[dB] + Gr[dB] - \sum Lx[dB] \tag{1.2}$$

In this equation, the received power $Pr$ $[dBm]$ is calculated based on the transmitted power $Pt$ $[dBm]$ that is delivered to the transmitting antenna; $Gt$ $[dB]$ and $Gr$ $[dB]$ are the transmitter and receiver antenna gains, respectively. The $\sum Lx$ $[dB]$ represents the total losses caused by the environment. The dB, or decibel, is a logarithmic unit to express the ratio between two values in the context of power used to describe gain or loss, whereas the dBm is an absolute measurement that indicates actual power with reference to 1 mW.

For our experiments, we use the FSPL channel model because the scenario involves a platoon traveling on a highway with no oncoming traffic or nearby buildings, and the inter-vehicle distances are relatively short. These are the conditions under which FSPL provides a reasonable first-order approximation. However, FSPL is a rather simplistic model of real radio propagation and does not account for essential effects such as ground reflections, multipath fading, and obstacle shadowing. For more realistic, higher-fidelity evaluations, propagation models such as the *Two-Ray Interference model*, *Obstacle Shadowing*, *Rayleigh*, and *Rician Fading* models should be considered in future experiments. These models capture reflections, scattering, and blockage effects that can significantly alter SINR, packet delivery, and the observable impact of jamming in real-world settings, and therefore have the potential to be incorporated in future studies.

#### 1.2.1.3 Wireless Communication Protocols

Wireless communication protocols play a vital role in enabling the real-time exchange of safety-critical information in connected and automated mobility systems. Protocols such as IEEE 802.11, originally designed for Dedicated Short Range Communications (DSRC), provide low-latency V2V links that support cooperative driving CCAM functionaly. Similarly, IEEE 802.11 (WiFi) is used for teleoperation and remote monitoring applications. Similarly, transport-layer protocols such as the User Datagram Protocol (UDP) [37] and the Secure Reliable Transport (SRT) [38] protocol are widely used for transmitting real-time sensor and video data over wireless networks. UDP offers minimal transmission delay by omitting reliability mechanisms such as acknowledgments or retransmissions, making it suitable for time-sensitive applications but vulnerable to packet loss. SRT, on the other hand, extends UDP with built-in encryption, error correction, and retransmission, improving reliability and confidentiality for real-time media streams. Encryption in these protocols typically relies on symmetric cryptographic algorithms [72] such as AES-128, AES-192, or AES-256, where higher key sizes provide stronger security at the expense of additional computational load and potential latency.

### 1.2.2 Communication Jamming Techniques

Communication jamming refers to the deliberate introduction of malicious signals or data intended to disrupt or corrupt legitimate wireless communication at the reciever. Such techniques can be employed both in real-world environments and simulation-based testbeds to verify and validate the safety and resilience of interconnected and automated vehicles against cybersecurity threats. Over the years, numerous researchers have investigated a wide variety of jamming techniques that differ in complexity, implementation requirements, and impact, depending on the characteristics of the target communication system. Some jamming techniques are relatively simple to design and deploy, while others demand detailed knowledge of the underlying communication protocols and physical-layer mechanisms.

A recurring challenge identified in the literature is inconsistent terminology, as similar jamming strategies are often referred to by different names across studies. To address this ambiguity and ensure consistent classification across our work, we adopt the taxonomy proposed by Lichtman et al. [11], which provides a structured, comprehensive framework for categorizing jamming and cybersecurity attacks.

Lichtman et al. [11] and other researchers [50], [73] categorize communication jamming techniques based on their implementation methods, signal characteristics, and intended effects. Below, we briefly introduce several representative jamming attacks, which are also summarized in Table 1.3.

**Barrage jamming** or **noise jamming** [11] floods a wide frequency band with additive white Gaussian noise (AWGN) to reduce the received SINR. Because it targets a broad spectrum, barrage jamming is effective against frequency-hopping systems. However, note that it is also possible to add noise to the specific channel or frequency if the jammer knows exactly which channel is used for data communication, thereby saving the jammer's resources. This type of jamming is also known as **spot jamming**.

**Partial-band** [73] jamming concentrates interference power over a selected subset of channels or frequencies, making it more energy-efficient than barrage jamming while still affecting multiple channels. It is particularly effective against communication systems that use frequency hopping, spread-spectrum, or multi-channel transmission, because it increases the probability of disrupting the receiver when it hops into the jammed portion of the frequency band.

In **Automatic Gain Control (AGC) [73] jamming**, the attacker transmits a strong, sudden, or high-power signal that forces the receiver's Automatic Gain Control (AGC) to reduce its gain rapidly. When the legitimate signal arrives, the receiver's gain is too low to properly amplify it, resulting in a distorted or unreadable signal.

In **equalization** [73] jamming, the attacker transmits crafted interference signals that often mimic channel characteristics, confusing or overloading the

Table 1.3: Summary of primary communication jamming techniques based on Lichtman et al.'s taxonomy [11]

| **Jamming Technique** | **Description** |
|---|---|
| Barrage Jamming | Jammer transmits high-powered noise-like energy across the entire portion of the spectrum occupied by the target. |
| Partial-Band Jamming | Jamming targets only a fraction of the total signal. |
| Automatic Gain Control (AGC) Jamming | Targets the receiver's AGC mechanism using a very low duty cycle but extremely high instantaneous power. |
| Equalization Jamming | Disrupts equalization mechanisms by targeting known pilot symbols used to estimate channel frequency response in multicarrier systems. |
| Synchronization Jamming | Disrupts synchronization signals used by the receiver to align in time and frequency. Requires high protocol awareness due to the sparsity of synchronization signals. |
| Nulling | Transmits a structured waveform that destructively interferes with the target signal ($\pi$-radian phase shift), effectively canceling it and leaving only channel noise in case the target signal is completely cancelled. |

equalizer. This prevents the receiver from accurately performing channel estimation and compensation, leading to the legitimate signal being misaligned, distorted, or unrecoverable. The examples include 'replay' or 'spoofing' cyber-attacks.

Most wireless systems rely on precise synchronization, such as symbol timing, carrier frequency alignment, or frame synchronization, to interpret received data. In **synchronization** [73] jamming, the attacker transmits signals, such as noise bursts, false preambles, or misleading synchronization patterns, that interfere with this process. As a result, the receiver fails to lock onto the correct timing or frequency, leading to symbol errors, dropped frames, or complete loss of communication.

In **nulling**, **cancellation**, or **destructive interference** [11] attacks, the attacker affects the signal destructively at the receiver. This is achieved by transmitting a signal identical to the target signal in time and frequency, but shifted in phase by 180 degrees. This requires precise synchronization and channel knowledge, but can effectively eliminate the legitimate signal at the receiver front end.

Litchman et al. classified jamming attacks according to their capabilities, including *time-correlation*, *protocol-awareness*, *learning*, and *spoofing* [11]. *Time-correlation* indicates the alignment of the jamming signal with the target signal in time. *Protocol awareness* is the attacker's understanding of the

Table 1.4: Summary of jamming strategies based on Vadlamani et al. [50]

| **Jamming Strategy** | **Description** |
|---|---|
| Constant | Continuously emits electromagnetic waves based on the random bit sequences to interfere with legitimate signal transmission until the end of an attack. It compromises the channel by creating persistent noise. |
| Deceptive | Emits a legitimate-looking signal to deceive the network, making it appear as though valid communication is occurring. |
| Random | Alternates between jamming and sleeping periods to conserve energy while still disrupting communication at irregular intervals. |
| Reactive | Monitors the channel and transmits interference only when it detects legitimate transmissions, making it efficient and difficult to detect. |

legitimate signal's protocol. In the context of *machine learning*, learning is the process by which an attacker derives knowledge from data. *Spoofing* involves impersonating legitimate signals to obtain unauthorized access. Among these five types of jamming attacks studied in this thesis, *DoS* and *barrage jamming* attacks are *non-correlated* and *non-protocol aware* and can, therefore, be carried out without detailed knowledge of the communication protocol and targeted signal. However, *destructive interference* and *deceptive jamming* require *time-correlated* and *protocol-aware* capabilities.

Vadlamani et al. [50] identify four principal jamming strategies that are widely examined in the literature:*constant jamming*, *deceptive jamming*, *random jamming*, and *reactive jamming*. These strategies emphasize the *operational behavior* of the jammer, how and when interference is introduced, thereby offering a complementary perspective to Lichtman et al.'s taxonomy, which classifies attacks based on the adversary's information and capabilities. A detailed comparison of these four jamming strategies is presented in Table 1.4.

### 1.2.3 Platooning Application

Platooning is a cooperative driving technology in which a group of vehicles, known as a platoon, travels closely together at high speeds, maintaining a small distance between each other. The vehicles in a platoon are equipped with advanced communication systems and cooperative cruise controllers that allow them to share information and coordinate their movements.

The attack-injection experiments described in this thesis have all been conducted using a platooning application available in the simulation-based Plexe framework. This platooning application is based on a CACC controller that implements the longitudinal control law described by Rajamani et al [74]. We refer to this controller simply as CACC in paper A and B. However, since

Plexe includes several types of CACC controllers, we decided to rename this controller to P1 (Plexe 1) in Paper C to avoid confusion with other controllers.

The P1 controller consists of an upper-level controller and a lower-level controller. The upper-level controller determines the desired acceleration for each vehicle in the platoon to maintain the desired spacing between the vehicles and ensure the platoon's string stability. String stability of a platoon refers to the ability of a group of coordinating vehicles to travel with the desired velocity and maintain a close distance to achieve the vehicle's on-road performance and efficiency [27]. A platoon has string stability if disturbances are not amplified when propagating along the vehicle string [39]. A string-stable platoon ensures that the following vehicles decelerate in a controlled manner without overreacting. If the platoon is string unstable, the second vehicle might brake harder than necessary, causing the third vehicle to brake even harder, potentially leading to a chain reaction of emergency braking or even collisions. The lower-level controller translates the desired acceleration into throttle and brake commands.

In the P1 controller, each vehicle receives information fromand preceding vehiclesng vehicle in the platoon via the wireless network. This information includes the controller's desired acceleration ($m/s^2$), the vehicle's actual acceleration ($m/s^2$), speed ($m/s$), XY position ($m$), and the time at which the data is measured ($s$). Below is the controller equation that computes the desired acceleration of the i-th vehicle in a platoon.

$$\begin{aligned}\ddot{x}_{i\_des} &= (1 - C_1)\ddot{x}_{i-1} + C_1\ddot{x}_l \\ &\quad - (2\xi - C_1(\xi + \sqrt{\xi^2 - 1}))\omega_n\dot{\epsilon}_i \\ &\quad - (\xi + \sqrt{\xi^2 - 1})\omega_n C_1(v_i - v_l) - \omega_n^2\epsilon_i\end{aligned} \tag{1.3}$$

$C_1$ is the weighting factor that takes on values between 0 and 1 where the default value is set to 0.5, $\xi$ is the damping ratio and can be set to 1 for critical damping, and $\omega_n$ is the controller's bandwidth, where the default value is set to 0.2Hz [6]. $\ddot{x}$ denotes the longitudinal acceleration of the vehicle where $\ddot{x}_{i-1}$ represents the acceleration of the preceding vehicle and $\ddot{x}_l$ represents the acceleration of the lead vehicle. Similarly, $v_i$ is the longitudinal velocity of the $i_{th}$ vehicle, and $v_l$ is the longitudinal velocity of the lead vehicle.

The first four terms of Equation 1.3 consist of information received from other vehicles via V2V communication. The fifth term consists of two parameters, $\omega_n^2$ and $\epsilon_i$, where $\omega_n^2$ is the control gain, and $\epsilon_i$ is the longitudinal spacing error of the $i$th vehicle, which is calculated using Equation 1.4, where $x_i$ is the position of the $i$th vehicle, $x_{i-1}$ is the position of the preceding vehicle, and L is the desired spacing.

$$\epsilon_i = x_i - x_{i-1} + L \tag{1.4}$$

The $\epsilon_i$ solely relies on sensor information acquired by each vehicle's own radar and is therefore unaffected by any communication loss. Note that the P1 algorithm mainly uses the radar to maintain consistent spacing between vehicles to ensure string stability. It does not use the radar to achieve collision avoidance, as is done in ACC algorithm. More details about setting the controller parameters, such as engine and driver parameters, can be found in the API section of the Plexe webpage [75]. In addition to this CACC algorithm, Plexe includes implementations of three other CACC algorithms which are known as ‘Flatbed’ [33], ‘Ploeg’ [34], and ‘Consensus’ [76].

### 1.2.4 Related Research

In the related research, Section 1.2.4.1 outlines the commonly accepted view of security benchmarking and summarizes prior work in this area. Section 1.2.4.2 presents studies on communication jamming techniques that are relevant to our work. In contrast, Section 1.2.4.3 describes several methods that could enable real-world communication jamming attacks. Section 1.2.4.4 reviews research that examines the impact of jamming attacks on platooning controllers, and Section 1.2.4.5 discusses approaches for enhancing the jamming resilience of CACC algorithms. Finally, Section 1.2.4.6 presents studies related to teleoperation applications and their vulnerability to intentional wireless interference.

#### 1.2.4.1 Security Benchmarking

Kanoun et al. [22] describe the key differences between benchmarking and other verification and validation techniques. Fundamentally, benchmarking is a technical agreement that defines the measures, the methods for obtaining the measures, and the context or domain in which those measures are considered valid. A dependability benchmark enables an objective characterization of system dependability and provides a fair basis for comparison between alternative systems or different versions of the same system. Benchmarking may also be applied to assess whether newer system versions maintain or improve dependability attributes [77] relative to previous releases.

According to Kanoun et al., benchmarking can rely on experimentation or system modeling. In experimental benchmarking, results are obtained from controlled experiments specified by the benchmark. These experiments typically involve applying a predefined workload or fault load to the system under test to obtain measurable indicators of performance or dependability. In the context of dependability benchmarking, fault loads are often introduced using fault-injection techniques that deliberately trigger controlled faults to evaluate the system’s fault tolerance and recovery mechanisms.

Researchers have proposed various security benchmarking frameworks across different cybersecurity domains. Oliveira et al. [26] introduced a two-phase benchmarking framework specifically for web service frameworks (WSFs), emphasizing security qualification and trustworthiness assessment. Similarly, Anisetti et al. [78] developed a security benchmark to evaluate the security assurance of OpenStack, an open-source cloud infrastructure. Additionally, Braun et al. presented NETCARBENCH [79], a benchmark designed to assess and compare techniques and tools for the development of in-vehicle communication networks. To the best of our knowledge, no prior research has established security benchmarks for the simulation-based assessment of CCAM systems.

#### 1.2.4.2 Communication Jamming Techniques

Researchers studying V2V communication systems have recognized the increasing challenges posed by cybersecurity threats and the importance of information security [12], [15], [29]. This section presents studies on communication jamming techniques that are relevant to our work.

Moser et al. [30] studied the impact of signal cancellation attacks where the attacker's signal interferes destructively with the legitimate signal. They demonstrated that the signal cancellation attack could effectively attenuate the signals up to 40 $dB$. Moser et al. demonstrated through their experiments that cancellation or destructive interference attacks are feasible, a finding that should be considered when assessing the security of advanced cooperative systems.

Clancy [32] studied the performance of OFDM transmission, *pilot jamming*, and *pilot nulling* attacks. According to the results obtained in this work, pilot jamming is roughly 2 $dB$ more efficient than barrage jamming, and pilot nulling is approximately 7.5 $dB$ more efficient than barrage jamming.

Mahal et al. [31] studied the impact of nulling attacks on cyclic prefixes in single-carrier frequency division multiple access (SC-FDMA) communication, which is employed for up-links in 4G and 5G mobile communication standards. Cyclic prefixes involve adding a copy of the end of a signal to the beginning of the signal to mitigate inter-symbol interference (ISI).

Patounas et al. [80] studied the prevention, detection, and mitigation of DoS attacks on IEEE 802.11p-based communication of a vehicle platoon. They implemented and tested intrusion-detection and handling mechanisms against barrage-jamming and data-falsification attacks.

#### 1.2.4.3 Real-world Communication Jamming Methods

Modern commercial and off-the-shelf hardware and software make it relatively easy for attackers to launch a wide range of jamming attacks against CCAM systems. In the following subsections, we briefly describe several representative methods that could enable such attacks.

**Software-Defined Radio (SDR):** SDR platforms are commonly accessible and can be used by attackers to generate arbitrary and malicious radio signals over broad frequency ranges, enabling the jamming of protocols such as GPS, IEEE 802.11p, and WiFi. For example, *LimeSDR* supports simultaneous transmission and reception from 10 MHz to 3.5 GHz with up to 30.72 MHz bandwidth [81], while *USRP B200/B210* offers a frequency range of 70 MHz to 6 GHz and bandwidth up to 56 MHz [82]. Additionally, the widely used *HackRF One* is a low-cost SDR capable of transmitting and receiving from 1 MHz to 6 GHz with up to 20 MHz bandwidth, making it suitable for introducing different types of communication jamming attacks [83]. Furthermore, these SDR devices can be paired with omnidirectional or directional antennas, including beam-forming antennas [84], to focus interference toward specific targets.

**Direct Digital Synthesis (DDS):** DDS [85] enables fast, real-time generation of analog waveforms by digitally constructing signals and converting them using a high-speed digital-to-analog converter. DDS modules can produce a wide range of frequencies, from a few hertz to several hundred megahertz. DDS is suitable for generating precise jamming waveforms such as narrowband interference. Standard DDS-based devices include the Analog Devices AD9957 [86] and AD9833 [87] waveform generators. DDS devices are practical options for portable and easily deployable jamming setups.

**Use of ML/AI-based Technologies for Cyberattacks:** With the rapid advancement and widespread availability of modern ML and AI technologies, attackers now have powerful tools to model and exploit wireless communication systems. Machine learning and AI can be used to analyze live wireless traffic patterns, learn channel behaviour, and infer when a receiver or transmitter is active. Such predictive capabilities enable attackers to implement more intelligent attack strategies, such as *reactive jamming*, in which interference is introduced only when legitimate signals are detected. This approach significantly reduces power consumption while increasing the attacker's stealth. Moreover, ML/AI-based jammers can adapt to dynamic channel conditions, such as variations in energy levels or timing patterns, allowing them to launch more targeted and effective jamming attacks [88], [89].

It is important to note that, under EU directives, manufacturing or using radio-frequency jammers to interfere with licensed public communication frequencies intentionally is illegal [90], [91], [92]. Even for testing purposes, the use of jammers, such as those that disrupt mobile or WiFi signals, requires special permits and must be carried out in strictly controlled facilities. Given these regulatory constraints, conducting jamming experiments in simulation environments or within controlled real-world setups (i.e., physical tesbeds, such as anechoic chambers), is highly advantageous. These environments allow safe and compliant testing while preventing unintended interference with external communication systems.

#### 1.2.4.4 Simulation-based Assessment of CACC Algorithms

Alipour-Fanid et al. [27] investigated the impact of the attacker's location when performing a reactive jamming attack on cooperative driving. They used a high-level model of the IEEE 802.11*p* protocol to study the impact of jamming attacks on a cooperative cruise controller implemented in MATLAB. They showed that targeting the vehicle behind the lead vehicle is most effective for an attacker to destabilize the platoon's string stability.

The study performed by the authors in [27] is similar to our work on jamming attacks against the IEEE 802.11*p* communication protocol, in which the same cooperative cruise controller is evaluated. Their flexible jamming model represents a wide range of jamming signal scenarios implemented in MATLAB.

We model and implement detailed *barrage* and *destructive interference* attacks at the physical layer of the communication system modeled in Veins.

Veins provides high-fidelity wireless communication models. Moreover, the attacker model implemented by Alipour-Fanid et al. is based on the Additive White Gaussian Noise (AWGN).

van-der Heijden et al. [13] proposed a novel attacker model and use it to evaluate the resilience and effectiveness of three cooperative cruise controllers provided in Plexe. One of these controllers is identical to the one evaluated in our work. Their results show that this CACC controller is highly sensitive to jamming attacks. Their work resembles ours in that they use the Plexe framework to conduct simulations. However, while they model the impact of jamming attacks as lost messages at the application level, we simulate the attacks at the physical layer.

Another important aspect of our simulations is granularity. The granularity of the attack parameter values in our test campaigns is relatively high, i.e., the step size of our attack model parameters was significantly smaller than those used in comparable studies [13], [27]. In addition, we classified the experimental results using deceleration profiles and collision incidents, whereas the other studies used speed profiles to classify the severity of the outcome. As part of future work, we plan to extend our classification scheme to enable direct comparisons with these results and other future studies.

#### 1.2.4.5 Improving Jamming Resilience of CACC Algorithms

The control algorithms of cooperative vehicles must be built resilient to jamming attacks to ensure the safety, operational continuity, security, and regulatory compliance of autonomous vehicle systems.

In a recent paper, Segata et al. [28] argue that no single communication technology can achieve the reliability required for advanced cooperative driving applications. Hence, they propose a fallback and recovery mechanism based on the assumption that future vehicles will be equipped with multiple communication interfaces, such as IEEE 802.11p, Visible Light Communication (VLC), and LTE-based Cellular V2X (C-V2X). This mechanism ensures that vehicles can safely transition to autonomous or manual driving. The authors show that the proposed fallback and recovery mechanism is feasible. However, designing such a system requires careful consideration, as poor design choices can lead to instability or even collisions.

van-der Heijden et al. [13] developed an evaluation framework for assessing the resilience of cooperative cruise controllers implemented in Plexe against jamming attacks. Based on their experimental findings, the authors suggested a graceful degradation from a cooperative cruise controller to an adaptive cruise control as a potential mitigation strategy.

Shahriar et al. [93] also proposed a synchronized braking mechanism in the cooperative cruise controller implemented in the Plexe simulation. This mechanism is a type of emergency braking that acts as a fail-safe to prevent rear-end collisions. They did not test their safety mechanism against jamming attacks. However, their focus is on avoiding rear-end collisions that could occur when braking due to the cooperative vehicles short inter-vehicle spacing.

In contrast to these contributions, our work extends the P1 (Plexe 1) [6] algorithm into two jamming resilient variants. We denote these extensions as *P1A* and *P1B*. The *P1A* extension incorporates a dedicated fallback controller that uses onboard radar measurements and increases the inter-vehicle spacing in case of communication losses. This design is further refined through two variants: *P1Aa*, which activates during complete communication failure, and *P1Ab*, which responds to partial communication loss while still attempting to preserve cooperative behaviour. The *P1B* extension adopts a simpler fallback approach by switching from *CACC* to standard *ACC* whenever message loss is detected.

#### 1.2.4.6 Evaluation of Teleoperation Applications

Researchers have previously investigated the impact of communication jamming attacks on *teleoperation systems*. In an early study, Hamdan and Mahmoud [94] showed that *communication delays* significantly undermine the stability of teleoperation systems. The authors of this study proposed the use of encryption algorithms to counter attacks on the wireless medium, although they noted that such cryptographic mechanisms remain ineffective against DoS attacks, highlighting the need for further empirical evaluation.

Bonaci et al. [95] evaluated the effects of *DoS* attacks of varying severity on teleoperated robotic surgery systems. Their findings are similar the results of Hamdan et al. by showing that *DoS* and *delay* attacks can severely degrade operator control and system stability. They concluded that while confidentiality and authentication techniques help protect against eavesdropping or message manipulation, they do not mitigate DoS attacks, underscoring the need for stronger resilience mechanisms in teleoperated systems.

The relevance of these studies lies in their evaluation of how packet *communication delays* and *DoS* attacks affect human-in-the-loop remote operation. However, the specific use cases differ from ours, as prior work primarily focused on bilateral teleoperation systems or surgical robots, whereas our study investigates remotely operated road vehicles (RORV).

A more closely aligned study with our work is presented by Rozsíval and Smrčka in [96], who evaluated a teleoperation setup similar to our system under test. They introduced NetLoiter, a tool designed to analyze RORV systems under different network conditions by directly interfacing with the TCP/UDP communication nodes, typically through a wired connection such as Ethernet. While this enables controlled manipulation of packet flows, the approach does not fully capture the wireless characteristics and variability encountered in real-world teleoperation scenarios.

In contrast, our work integrates a complete wireless communication stack into the evaluation environment to more accurately reflect real deployment conditions. This allows the RORV system to interact with the communication layers exactly as it would during actual remote operation, enabling more realistic end-to-end testing of teleoperation behaviour under jamming attacks and wireless impairments.

## 1.3 Summary of Appended Papers

In this section, we summarize all the publications included in this thesis. Fig. 1.5 provides an overview of these publications.

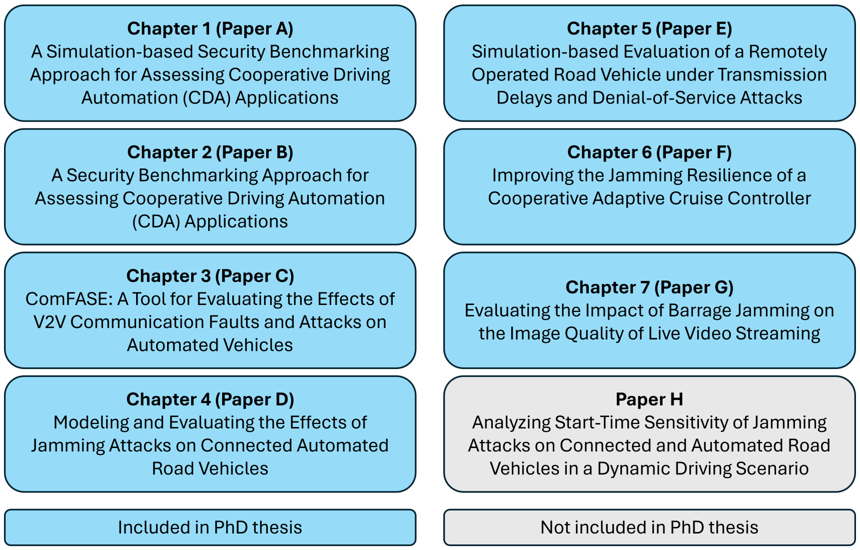
**Chapter 1 (Paper A)**
A Simulation-based Security Benchmarking Approach for Assessing Cooperative Driving Automation (CDA) Applications
**Chapter 2 (Paper B)**
A Security Benchmarking Approach for Assessing Cooperative Driving Automation (CDA) Applications
**Chapter 3 (Paper C)**
ComFASE: A Tool for Evaluating the Effects of V2V Communication Faults and Attacks on Automated Vehicles
**Chapter 4 (Paper D)**
Modeling and Evaluating the Effects of Jamming Attacks on Connected Automated Road Vehicles
**Chapter 5 (Paper E)**
Simulation-based Evaluation of a Remotely Operated Road Vehicle under Transmission Delays and Denial-of-Service Attacks
**Chapter 6 (Paper F)**
Improving the Jamming Resilience of a Cooperative Adaptive Cruise Controller
**Chapter 7 (Paper G)**
Evaluating the Impact of Barrage Jamming on the Image Quality of Live Video Streaming
**Paper H**
Analyzing Start-Time Sensitivity of Jamming Attacks on Connected and Automated Road Vehicles in a Dynamic Driving Scenario
Included in PhD thesis
Not included in PhD thesis

Figure 1.5: Overview of the research publications.

### 1.3.1 Paper A. Simulation-based Security Benchmarking Approach for Assessing Cooperative Driving Automation (CDA) Applications

The work presented in this paper is intended as an initial contribution towards a definition of *security benchmarks* for simulation-based assessment of CDA applications concerning their ability to operate safely in the presence of jamming attacks. In general, the primary motivation for defining benchmarks for computer-based systems is to provide a widely accepted and easy-to-use procedure for evaluating or comparing system implementations, components, or design solutions. Regarding basic concepts and main objectives, security benchmarking is closely related to dependability benchmarking.

Since security benchmarking of CDA applications is a novel topic, we would like to emphasize that our benchmarking framework is intended as a tentative example of how security benchmarks for assessing the jamming resilience of a CDA application could be defined. This is not intended as a final solution but as a starting point for a broader effort to develop security benchmarks for CDA applications, including benchmarks for attacks beyond jamming.

The core components of our proposed security benchmark are the driving scenario and the attack model. To illustrate the role these components would play in future definitions of jamming resilience benchmarks, we utilized two driving scenarios, braking and sinusoidal, as stimuli to evaluate the robustness of a platooning application. In addition, we injected barrage-jamming attacks into the vehicle communication system using the IEEE 802.11p protocol. Other system components influencing the evaluation, such as the wireless communication model, wireless channel model, and the number of vehicles, are kept constant throughout the testing and evaluation process.

We demonstrate that barrage jamming attacks can compromise safety, leading to emergency braking and collisions among platooning vehicles. Our findings also indicate that the severity of barrage jamming attacks varies with driving scenario, with the most severe impacts, such as collisions, occurring when the attack is initiated during vehicle acceleration. This outcome is closely tied to the design of the CACC controller model and explains why the platooning system is more vulnerable to attacks during acceleration. During these phases, vehicles rely more on timely and accurate inter-vehicle communication to maintain string stability and desired inter-vehicle gaps. Communication degradation caused by jamming therefore propagates rapidly through the controller, amplifying transient disturbances. As a result, even short-duration attacks can trigger abrupt control responses, such as emergency braking, that increase the likelihood of rear-end collisions.

When utilizing the specific CACC controller [28], the lead vehicle periodically sends acceleration and deceleration commands to the platoon's following vehicles. In case of communication loss, the following vehicles continue accelerating, decelerating, or keeping a constant speed according to the last received command. If a jamming attack begins to block the communication channel during an acceleration period, the affected vehicles will continue to accelerate and cause collision when the lead vehicle decelerates.

The *attack start-time* is not the only attack parameter that influences the likelihood of a collision. The *attack duration* and *attack value* are other attack parameters that influence the outcome. The longer attacks are generally more likely to cause a collision. However, attack durations longer than a certain threshold do not significantly increase the number of severe outcomes. Higher attack values lead to greater signal distortion, which can eventually cause communication failure. This loss significantly contributes to collisions when vehicles accelerate. We observe fewer collisions for attacks initiated when the vehicles are braking because the communication loss happens already when the vehicles have started to reduce their speed.

### Statement of Contribution

This is collaborative work with my supervisors, Behrooz Sangchoolie and Johan Karlsson. I was responsible for developing the concept of the security benchmarking framework, integrating it with my research topic, conducting test campaigns, performing a literature review, and analyzing experimental outcomes. Additionally, I took the lead in writing the paper. I would also like to highlight that our paper was recognized as one of the best accepted and presented papers at the conference. Overall, My supervisors ensured that the quality of the paper met the required academic standards.

### 1.3.2 Paper B. A Security Benchmarking Approach for Assessing Cooperative Driving Automation (CDA) Applications

This paper extends our previous work, in which we introduced a security benchmark to evaluate the resilience of CCAM systems against communication jamming attacks. In this work, we introduce a structured framework for defining a security benchmark. The framework emphasizes repeatability, objective comparison, and practical applicability. It provides a structured approach for evaluating cybersecurity attacks and assessing system behavior through performance indicators such as comfortable braking, emergency braking, and collision incidents.

In this study, we evaluate four CACC algorithms: *P1*, *Flatbed*, *Ploeg*, and *Consensus* under barrage-jamming attacks using detailed simulations. The results reveal significant differences in how each CACC algorithm responds to jamming attacks. The Consensus algorithm demonstrates the highest level of resilience, mainly because it relies on local onboard sensors and integrates data from all vehicles in the platoon. In contrast, the P1 algorithm is the most vulnerable, primarily because it depends almost entirely on V2V communication, which is directly targeted by jamming attacks. An important finding is that CDA systems are most susceptible to jamming attacks during acceleration phases, where vehicle spacing and synchronization are more critical and thus more likely to be disrupted.

The contribution of this study lies in both the development of the benchmarking framework and its application to CACC algorithms, providing a method for assessing the cybersecurity resilience of CDA applications against attacks. In conclusion, this paper presents a valuable approach for evaluating and comparing the jamming resilience of cooperative vehicle automation algorithms. Systematically identifying the strengths and vulnerabilities of different algorithms under controlled attack scenarios helps develop more secure and dependable automated driving systems. The proposed benchmarking framework is adaptable to various CACC algorithms and offers a foundation for broader investigations into the cyber-physical resilience of future cooperative and connected mobility infrastructures.

#### Statement of Contribution

This paper builds on our previous work by extending the development and application of a security benchmarking framework for cooperative driving automation systems. The research was carried out collaboratively with my supervisors, Behrooz Sangchoolie and Johan Karlsson. My main contributions included refining the conceptual design of the benchmarking framework, executing the test campaigns, and analyzing the outcome. I also took the lead in drafting and writing the manuscript. My supervisors helped ensure that the paper met high academic and technical standards.

### 1.3.3 Paper C. ComFASE: A Tool for Evaluating the Effects of V2V Communication Faults and Attacks on Automated Vehicles

In this paper, we introduce ComFASE, a versatile fault and attack simulation engine for studying consequences and safety implications of communication failures in interconnected automated vehicular systems. The tool is flexible in modelling different types of faults and attacks that may compromise the reliability of wireless messages. It enables detailed simulations to assess the safety implications of cybersecurity attacks and communication faults in realistic traffic scenarios. To this end, ComFASE utilizes four existing simulation environments: Plexe (for platooning simulation), Veins (a vehicular network simulator), SUMO (a traffic simulator), and OMNeT++ (a network simulator).

The tool supports automated fault- and attack-injection test campaigns. The campaign run is divided into three phases: configuration, execution, and result classification. During configuration, the user defines a traffic scenario, sets various communication model parameters, and provides a campaign vector. The traffic scenario can be tuned with respect to various system parameters dealing with road conditions, vehicle features, system size, scenario maneuvers, and simulation time. ComFASE currently supports simulation of the physical (PHY) and media access (MAC) layers of the IEEE 802.11p standard for wireless access in vehicular environments (WAVE). The user can configure the communication model by selecting one of three wireless channel models, the packet size, and the beaconing period.

The campaign vector includes information on the selected attack model, the vehicles to be attacked, and the attack model parameters. The latter consists of the attack start time, attack duration, and attack value. The attack's start time and duration are defined relative to the traffic scenario's time axis, whereas the attack value depends on the attack model. Another essential part of the configuration phase is the execution of the golden run, which generates a profile of the system's behavior under fault-free circumstances. The data collected during the golden run is later used to classify the outcomes of the attack simulations.

During execution, ComFASE automatically runs the test campaign defined by the configuration data. During the simulations, data is collected from SUMO about the movements of the vehicles in the investigated system, including velocities, accelerations, decelerations, and collision incidents. In the result classification phase, automated analyses classify the outcomes of attack simulations by severity. These analyses compare the target system's behavior under attack with its behavior in an attack-free run.

To demonstrate the tool's functionality and applicability, we present results from a series of simulation experiments in which we injected delay and denial-of-service attacks into wireless messages exchanged between vehicles in a platooning application. The results show how different attack variants and attack model parameters influence the platooning system, leading to collision incidents, benign outcomes, negligible outcomes, and non-effective outcomes.

**Statement of Contribution**

This work was conducted with my co-author, Mehdi Maleki. Together, we developed the ComFASE tool, configured the test campaigns, and performed the analysis. I led the conceptual development of the study, formulated the main ideas, and took primary responsibility for writing the manuscript. My supervisors, Behrooz Sangchoolie and Johan Karlsson, provided valuable feedback throughout the process, strengthening both the technical content and the presentation's clarity.

### 1.3.4 Paper D. Modeling and Evaluating the Effects of Jamming Attacks on Connected Automated Road Vehicles

In this paper, we propose and utilize simulation models to examine the impact of three types of jamming attacks: *destructive interference*, *barrage jamming*, and *deceptive jamming.* The primary objective of this study is to evaluate the impact of these attacks on vehicle safety by analyzing vehicle collision incidents and the deceleration profiles. Our findings reveal that jamming attacks pose significant risks to the stability and safety of platooning equipped with CACC algorithms that rely solely on communication and lack fallback mechanisms.

We conducted three attack-injection campaigns to evaluate the impact of *destructive interference* attacks on platooning behavior. In the first campaign, the attack targeted vehicle 2 (the vehicle behind the leader); in the second, it targeted vehicle 4 (the last vehicle in the platoon); and in the third, all vehicles were targeted simultaneously. When all vehicles were attacked, 27.5% of the experiments resulted in collisions. Targeting only vehicle 4 led to collisions in 26% of the cases, while targeting vehicle 2 resulted in collisions in 7% of the cases.

We observed that vehicle 4 was significantly more vulnerable to destructive interference attacks than vehicle 2. This high vulnerability is primarily due to the distance between the target and the leader vehicles. In this study, we used the Free Space Path Loss (FSPL) model, in which signal attenuation depends strongly on the distance between the transmitter and receiver. Being the farthest vehicle from vehicle 1, vehicle 4 experiences significant signal attenuation, making it particularly susceptible to the attacks.

We also conducted barrage-jamming attacks on all vehicles in the platoon, with 48% of the experiments resulting in collisions. To better understand the impact of barrage-jamming attacks, we identified the vehicles responsible for the collisions. Our analysis revealed that vehicles 2, 3, and 4 accounted for 41%, 43%, and 16% of the collisions, respectively. This outcome highlights how barrage-jamming attacks can disrupt the coordination and safety of the platoon. We also injected deceptive jamming attacks where 47% of the total experiments resulted in collisions.

These results, in which many experiments resulted in collisions, emphasize the need for robust error-handling mechanisms in CACC controllers to mitigate the risks posed by jamming attacks. Our findings suggest that the current implementation of the CACC model we tested lacks sufficient resilience against message loss due to jamming attacks. By demonstrating the impact of these attacks in a controlled simulation environment, the study underscores the importance of evaluating platooning applications under jamming attacks. This work provides valuable insights for designing and developing more secure and resilient communication protocols and control algorithms for connected automated vehicles.

**Statement of Contribution**

This paper is a collaborative effort with my co-author, Mehdi Maleki. Mehdi Maleki served as the first author and contributed to the development of the tool. I led the project in conceptualizing the idea, conducting the literature review, and focusing particularly on attack modeling. Additionally, I took the lead role in writing the paper. I received valuable feedback from my supervisors, Behrooz Sangchoolie and Johan Karlsson, which helped to enhance the quality of this manuscript. I am also proud to mention that this paper received the second-best paper award from the PRDC program committee in 2022.

### 1.3.5 Paper E. Simulation-based Evaluation of a Remotely Operated Road Vehicle under Transmission Delays and Denial-of-Service Attacks

This paper presents a comprehensive study of the jamming resilience of a remotely operated road vehicle (RORV) under wireless communication disruptions. We focus on evaluating two safety mechanisms: *safe braking* and *disconnection*, implemented in the RORV's control software to ensure secure and dependable operation under communication jamming attacks. Remotely operated vehicles, which are increasingly used in hazardous or restricted environments such as mines and harbors, rely heavily on wireless communication links between the vehicle and a remote operator. However, this dependency makes them susceptible to latency, interference, and DoS attacks, potentially compromising safety. To address these concerns, we employed the 'SiL' testing approach. We use the ComFASE tool, an open-source framework for injecting faults and attacks into vehicular networks. ComFASE was extended in this study to support WiFi communication through the Veins INET framework, allowing the integration of control software from the RORV system into our simulation environment.

The RORV system tested in this work, developed by a company called 'Roboauto', comprises three main components: the *electronic control unit* (ECU), the *gateway*, and the *remote station* (RS), along with a simulated vehicle dynamics model. The ECU exchanges control and video data with the RS over UDP communication channels, while safety mechanisms within the ECU continuously monitor network performance. The safe braking mechanism activates when communication delays exceed 150 ms, applying full braking until the delay returns to normal levels. If no communication is detected within 1500 ms, the disconnection mechanism initiates, safely stopping the vehicle and terminating the connection. To test these mechanisms, we conducted extensive experiments, 1680 involving packet transmission delays and 23 involving DoS attacks, by varying parameters such as delay duration, intensity, and attack start time. The results were classified into four categories based on the vehicle's speed profile: *Normal*, *speed deviation*, *safe braking*, and *disconnection*.

The results revealed that the safety mechanisms functioned effectively in most scenarios. Approximately 45% of delay-based experiments resulted in disconnection, 31% triggered safe braking, 7% showed minor speed deviations, and 16% exhibited no noticeable impact. Communication delays below 150 ms had no effect, while delays between 350 ms and 1500 ms typically triggered safe braking, and those exceeding 1500 ms led to disconnection. All DoS attacks caused disconnection within about three seconds of activation, confirming the system's consistent response to total communication loss. However, some inconsistencies were observed in the timing of safety mechanism activation, where triggers occurred slightly earlier or later than expected. These discrepancies were attributed to processing latencies or transient spikes in communication delay introduced by the simulation environment.

A threat to validity we identified for this study is the synchronization between the real-time execution of the RORV software components and the simulation environment. While the RORV software operates in real time, the Veins INET simulation framework inherently runs at a much faster pace. This discrepancy led to timing mismatches between the simulated communication events and the software modules' real-world timing expectations. Moreover, the scheduler struggled to maintain stable timing consistency under high network load or when multiple modules exchanged messages at short intervals, leading to delays or premature timeouts in the software. Authors in [97] also highlight the fundamental challenges of ensuring **timing determinism**, that is, the ability of a system to execute actions at precise and predictable times, **low intrusiveness**, that is related to introducing minimal disturbance to the system under test, and correct **temporal behavior** to preserve the timing relationships between events. The authors show that even minor, unintended timing deviations, introduced by schedulers, hardware layers, or software-based injection, can lead to non-representative system behavior.

To mitigate this limitation, we systematically analyzed the timing behavior of both the simulated and real systems and derived an empirical equivalence mapping between simulation time and real time. By calibrating simulation steps and timing parameters according to this mapping, we ensured that simulated communication delays, message arrivals, and system responses more closely align with the real-time execution of the RORV components.

In conclusion, the study demonstrated that the tested safety mechanisms are highly effective at mitigating the adverse effects of communication failures in remotely operated vehicles. The SIL-based approach proved valuable for early validation of software safety functions without the need for physical testing, providing an efficient means to identify potential vulnerabilities. In the future, we will focus on improving simulation fidelity, addressing timing deviations, and expanding ComFASE's capabilities to handle more complex network conditions.

**Statement of Contribution**

This paper is the outcome of a collaborative effort with my co-authors Maytheewat Aramrattana, Mehdi Maleki, Peter Folkesson, Behrooz Sangchoolie, and Johan Karlsson. Maytheewat made significant contributions by integrating the ComFASE tool with Roboauto's SiL components and the Veins INET framework, which models WiFi communication systems. I led the work across multiple technical and coordination fronts, including in-depth technical discussions with Roboauto engineers, the selection and modeling of jamming attack scenarios, and the configuration and execution of attack simulations. I also took the lead in drafting and refining the manuscript. Throughout the research process, I received valuable guidance and constructive feedback from my supervisors and my colleague, Peter Folkesson, which greatly enhanced the technical depth, scientific rigor, and overall clarity of the final publication.

### 1.3.6 Paper F. Improving the Jamming Resilience of a Cooperative Adaptive Cruise Controller

This paper presents a comprehensive simulation-based study on the resilience of several CACC algorithms against barrage jamming. CACC extends Adaptive Cruise Control (ACC) by enabling platoons of vehicles to exchange information wirelessly, maintaining string stability and safe inter-vehicle distances. We evaluate four existing CACC algorithms, *Ploeg*, *Consensus*, *Flatbed*, and the first *CACC* implemented in Plexe that we call Plexe 1 (P1). Ploeg and Consensus follow the Constant Time Headway (CTH) policy, whereas *Flatbed* and *P1* follow the Constant Vehicle Spacing (CVS) policy. The CTH-based algorithms primarily rely on local sensors, while the CVS-based algorithms depend heavily on V2V communication, making them potentially more vulnerable to jamming.

Simulation results confirm that CTH-based algorithms such as *Ploeg* and *Consensus* are pretty resilient to *barrage jamming*, and able to maintain string stability. In contrast, CVS-based algorithms, including P1, are highly vulnerable because they lack fallback mechanisms to handle communication failures. However, since CVS experiences higher traffic and tighter platooning, we explore ways to enhance its resilience. To this end, we propose two new extensions to the *P1* algorithm, P1A and P1B.

The *P1A* algorithm incorporates a fallback mechanism that uses onboard radar data and increases inter-vehicle spacing when communication failures are detected. Two variants, *P1Aa* and *P1Ab*, are introduced to handle complete and partial communication losses, respectively. Meanwhile, *P1B* employs the standard *ACC* as a fallback algorithm, automatically switching from *CACC* to *ACC* when message loss is detected. The fallback activation in both extensions is based on timestamp checks that determine whether the latest messages have been received within the beaconing interval of *100* ms.

We use ComFASE to carry out the attack injection simulations. Out of 3575 simulations, the baseline *P1* algorithm experienced 1475 collisions, confirming its poor resilience to jamming. In contrast, *P1Ab* reduced collisions dramatically, eliminating the collisions within specific noise power ranges and cutting them by up to 66% in higher noise conditions. *P1B* achieved even higher resilience, preventing collisions under both partial and total communication losses by gracefully switching to *ACC* control.

Comparative analysis across all four CACC algorithms revealed that the *Consensus* and *Ploeg* controllers were inherently more robust to jamming due to their reliance on local sensors and adaptive headway policies. However, the enhanced *P1B* algorithm outperformed all others in terms of strong resilience to communication failures. These results demonstrate that carefully designed fallback mechanisms can significantly strengthen the robustness of CVS-based CACC algorithms without sacrificing their efficiency advantages.

Overall, this study provides an essential step toward building more secure, reliable, and resilient cooperative driving systems that can withstand communication jamming attacks. The paper concludes that achieving true resilience in cooperative driving systems requires a cyber-physical co-design approach that combines communication-level defenses, such as frequency hopping and adaptive channel selection, with control-level strategies, including adaptive spacing and graceful degradation of *ACC*.

#### Statement of Contribution

The co-authors of this paper are Ludvig Ohlsson, Karthik Sharma, Behrooz Sangchoolie, and Johan Karlsson. Together with Ludvig Ohlsson and Behrooz Sangchoolie, I contributed to the development of the extensions to the P1 algorithm. In collaboration with Ludvig and Karthik, I configured attack scenarios, ran attack simulation campaigns, and ensured the experimental setup was accurate. Moreover, I led the overall work, including an in-depth analysis of the experimental results, writing, and refining the paper with my supervisors' input, which significantly enhanced the technical correctness, coherence, and clarity of the final publication.

### 1.3.7 Paper G. Evaluating the Impact of Barrage Jamming on the Image Quality of Live Video Streaming

Wireless communication plays a critical role in modern safety-critical Connected, Cooperative, and Automated Mobility (CCAM) systems. As reliance on wireless communication systems increases, so does the exposure to communication jamming attacks that can disrupt connectivity and compromise the integrity and availability of wireless communication. Among the various communication jamming attacks, such as *pilot jamming*, *deceptive jamming*, *destructive interference*, and *barrage jamming*, we used barrage jamming because it is easy to execute, as it requires minimal knowledge of the target communication system properties, such as the communication protocol, carrier frequency, modulation scheme, and receiver's sensitivity.

We assessed the impact of barrage jamming on video streams transmitted over WiFi using two communication protocols: *User Datagram Protocol (UDP)* and *Secure Reliable Transport (SRT)*. For SRT, we further evaluated three configurations: unencrypted, AES-128, and AES-256 encryption. Instead of focusing solely on complete denial-of-service, our analysis targeted partial communication degradation, as even moderate quality loss can mislead functions that rely on wireless data, such as perception and object-detection algorithms. We quantify video quality degradation using two objective quality metrics, *PSNR* and *MS-SSIM*, which enable consistent comparisons across protocols and encryption modes.

To demonstrate practical relevance, we evaluated a *drone surveillance* subsystem within the CCAM system, consisting of a 'surveillance drone' and an 'automated shuttle' specialized to deliver logs from the forest [98]. The drone provides critical information about non-line-of-sight (NLOS) hazards beyond the reach of an automated shuttle's onboard sensors via live video streaming. Video degradation caused by natural or intentional interference in such scenarios can compromise hazard detection and endanger system safety. To evaluate jamming resilience, we developed a physical testbed to conduct controlled jamming attack campaigns. We used a radio-frequency (RF) anechoic chamber, which provides a repeatable, interference-free environment ideal for precise and reliable evaluation.

The study reveals that *UDP* provides the highest resilience to jamming, maintaining better video quality and experiencing less permanent frame loss than SRT. Although SRT produced more perfect (undistorted) frames, it was more sensitive to interference, especially when encryption was enabled. Encrypted SRT variants showed increased frame loss and fewer high-quality outcomes due to the added computational overhead and stricter packet-handling requirements. The work highlights necessary trade-offs: *UDP* offers robustness and low latency but no security guarantees, whereas *SRT* provides security and privacy under attack-free conditions, but becomes more sensitive to jamming attacks, particularly when encryption is used.

Overall, the results show that selecting a protocol for safety-critical video transmission requires balancing latency, robustness, reliability, and confidentiality. The paper concludes by outlining future work, including evaluations under more diverse jamming conditions, additional protocols and encoding formats, alternative encryption schemes, and the use of non-reference video-quality metrics to better approximate real-world attack scenarios. Such extensions will enable a more comprehensive assessment of protocol resilience across varying operational conditions.

Note that in this study, we focus exclusively on evaluating the jamming resilience of the video stream and the communication protocols. Other CCAM functionalities, such as object detection and classification, and other safety functions that depend on the received video, are beyond the scope of this work and are planned for future work. Moreover, the attack parameter values used in this study are specific to the controlled physical setup. For example, the 19 dB attack value is valid only for the fixed experimental configuration employed. This value may change if other parameters, such as the distance between the transmitter and receiver, the attack duration, or the attack start time, are modified. Adjusting any of these can alter the system's jamming strength. The same applies to setting any other parameter.

### Statement of Contribution

In this work, I worked with my co-authors Peter Folkesson, Behrooz Sangchoolie, and Md Masoom Rabbani. Together with my colleague Peter, we developed a physical testbed to conduct communication-jamming attacks. Peter made a significant contribution by automating the jamming experiments, ensuring repeatability and precision in the test process. I designed and implemented the attack scenarios, conducted in-depth data analysis, and interpreted the experimental results. Behrooz Sangchoolie provided valuable support and detailed insights during the analysis phase, helping to refine the interpretation of key findings. I also contributed to the writing of the paper while receiving constructive feedback and guidance from co-authors and my supervisors.