A Security Benchmarking Approach for Cooperative Driving Automation (CDA) Applications

Downloaded from: https://research.chalmers.se, 2026-11-18 17:16 UTC

Citation for the original published paper (version of record):
Malik, M., Sangchoolie, B., Karlsson, J. (2025). A Security Benchmarking Approach for Cooperative Driving Automation (CDA) Applications. Mobile Networks and Applications, 30(3-4): 356-368. http://dx.doi.org/10.1007/s11036-025-02460-x

N.B. When citing this work, cite the original published paper.

research.chalmers.se offers the possibility of retrieving research publications produced at Chalmers University of Technology. It covers all kind of research output: articles, dissertations, conference papers, reports etc. since 2004. research.chalmers.se is administrated and maintained by Chalmers Library

(article starts on next page)

Mobile Networks and Applications (2025) 30:356–368
https://doi.org/10.1007/s11036-025-02460-x

RESEARCH

A Security Benchmarking Approach for Cooperative Driving Automation (CDA) Applications

Mateen Malik[1,2] · **Behrooz Sangchoolie**[1] · **Johan Karlsson**[2]

Accepted: 25 August 2025 / Published online: 22 September 2025
© The Author(s) 2025

Abstract
This paper proposes a reference model for defining security benchmarks for the safety assessment of Cooperative Driving Automation (CDA) applications. Our reference model provides a systematic approach to benchmark the resilience of CDA applications against malicious attacks through extensive system simulations. It enables the test repeatability and comparison of results across different implementations of CDA applications. In our approach, a benchmark is defined as a series of tests that expose the target system to specific attacks while recording its response. Using this model, we define a benchmark for evaluating the resilience of Cooperative Adaptive Cruise Control (CACC) algorithms against barrage jamming attacks targeting the physical layer of the IEEE 802.11p communication standard. We apply this benchmark to assess and compare the performance of four CACC algorithms: P1, Flatbed, Ploeg, and Consensus. The benchmark measures reveal that the Consensus algorithm demonstrates the highest resilience against jamming attacks, primarily due to its heavy reliance on onboard sensors and the use of sensor data from all other vehicles for decision-making. In contrast, the P1 algorithm, which depends mainly on vehicle-to-vehicle (V2V) communication, proves to be the most vulnerable. Furthermore, the results indicate that vehicles are most susceptible to jamming attacks during acceleration phases, making these periods critical for security evaluation. These findings validate the effectiveness of our benchmarking framework in identifying strengths and vulnerabilities of CACC algorithms under cyberattacks.

Keywords Reference modeling · Security benchmarks · Cooperative driving automation · Simulation-based testing · Jamming attacks · Platooning system

1 Introduction

The advent of wireless communication standards for the automotive sector has paved the way for the development of cooperative driver automation (CDA) systems [1]. A key feature of these systems is that they utilize vehicle-to-vehicle (V2V) and vehicle-to-infrastructure (V2I) wireless communication to provide advanced vehicle control functions.

A widely researched example of a CDA application is vehicle platooning, which is regarded as a promising solution to make road transportation more comfortable, fuel-efficient, and safer. A platoon is a virtual train, or string, of vehicles consisting of a leader and an arbitrary number of trailing vehicles that coordinate their movements via an exchange of wireless messages to improve traffic flow. By enabling short inter-vehicle distances at high speeds, platooning can effectively reduce drag and thereby fuel consumption [2, 3].

Platooning systems [4, 5] utilize a concept known as Cooperative Adaptive Cruise Control (CACC) for the longitudinal control of vehicles. CACC is an improvement of Adaptive Cruise Control (ACC) found in many contemporary road vehicles. One challenge in designing CACC algorithms is that they must be highly resilient to various types of security attacks, including internal attacks originating

✉ Mateen Malik
mateen.malik@ri.se

Behrooz Sangchoolie
behrooz.sangchoolie@ri.se

Johan Karlsson
johan@chalmers.se

[1] Dependable Transport Systems, RISE Research Institutes of Sweden, Borås, Sweden

[2] Computer Science and Engineering, Chalmers University of Technology, Gothenburg, Sweden

from within a compromised vehicle and jamming attacks directed at the wireless medium.

Despite the large body of research published over the last three decades on the construction of platooning systems, relatively few studies have focused on assessing the resilience of CACC algorithms against malicious attack [5, 6]. These studies are primarily conducted through system simulations, as purely control theoretical approaches are challenging to pursue when analyzing the impact of cyberattacks on distributed control systems.

Unfortunately, it is often difficult to replicate and compare the results of such simulation studies, as different research groups tend to use slightly different assumptions when designing and configuring their experiments. Additionally, the descriptions of the experiments sometimes lack the necessary details to replicate the experiments and compare the results. To provide a first step towards avoiding such difficulties, we propose in this paper a reference model for benchmarking the resilience of CACC algorithms against security attacks through simulations.

To illustrate the application of the reference model, we use it to develop a concrete benchmark for assessing the resilience of CACC algorithms against barrage jamming attacks. We then employ this benchmark to assess and compare the resilience of four previously published CACC algorithms against barrage jamming attacks.

To this end, we run simulations using a tool called ComFASE [7], which automates the simulation of multiple attacks. ComFASE runs on top of a collection of open-source simulators: Plexe (a cooperative driving framework) [8]], Veins (a vehicular network simulator) [9], OMNet++ (a network simulator) [10], and SUMO (Simulation of Urban MObility) [11].

The work presented in this paper builds upon the findings reported in our earlier work [12]. In that work, we introduced a conceptual discussion and outlined a direction for the development of a security benchmarking framework for CDA systems. There, we evaluated the jamming resilience of a single CACC algorithm [8] across two distinct driving scenarios. In contrast, this paper makes the following contributions:

1. A proposal for a reference model to support the definition of security benchmarks.
2. Based on the proposed reference model, a definition of a concrete benchmark for assessing the resilience of CACC algorithms against barrage jamming attacks.
3. Use of the proposed benchmark for a comparative assessment of four previously published CACC algorithms: P1[1] [8], Flatbed [13], Ploeg [14], and Consensus [15].

[1] We call the first CACC algorithm available in Plexe as P1.

The remainder of the paper is organized as follows. Section 2 describes related work on security benchmarking, CACC algorithms, and investigations of the impact of jamming attacks on platoons. Section 3 presents the reference model. Section 4 describes the proposed benchmark for assessing the resilience of CACC algorithms against barrage jamming attacks. The benchmark results are presented in Section 5. In Section 6, we present general observations based on the benchmark results and discuss potential threats to their internal and external validity. Section 7 concludes the paper with a summary of findings and future research directions.

2 Related work

2.1 Security benchmarks

Benchmarking has proven to be pivotal for advancing the state of the art in certain fields of computer engineering, such as computer architecture [16]. However, defining a benchmark, or a benchmark suite, is a challenging undertaking that requires involvement and interaction among many groups of researchers, developers, and other stakeholders in the field of interest.

Researchers have proposed security benchmarking frameworks for various cybersecurity domains. Oliveira et al. [17] introduced a two-phase benchmarking framework for web service frameworks (WSFs), focusing on security qualification and trustworthiness assessment. Similarly, Anisetti et al. [18] developed a security benchmark for assessing the security assurance of OpenStack, an open-source cloud infrastructure. Braun et al. also presented NETCARBENCH [19], a benchmark for assessing and comparing techniques and tools used to design in-vehicle communication networks. To the best of our knowledge, our previous work [12] was the first to introduce the initial concept and discuss the importance of security benchmarking for evaluating CDA applications.

2.2 CACC algorithms

The main advantage of CACC algorithms is to enhance vehicle automation and safety. They do this by enabling vehicles to maintain optimal spacing, speed, and coordination by using onboard sensors and V2V communication [20].

The CACC algorithms investigated in this work employ two distinct control policies to maintain the distances between vehicles. These policies are *Constant Vehicle Spacing (CVS)* and *Constant Time Headway (CTH)* [21]. CVS maintains a fixed distance between vehicles regardless of their speed, while CTH adjusts the spacing based on the vehicle's velocity. Faster speeds require larger gaps to

maintain the safety and stability of interconnected vehicles, such as those in a platoon.

The CACC (P1) algorithm developed by Segata et al. [8] is designed to maintain a constant inter-vehicle distance using a CVS policy [21]. To achieve this, each vehicle relies on speed and acceleration data from both preceding and leading vehicles. The foundational design of this algorithm is established by Rajamani et al. [22].

Ali et al. [13] introduced the Flatbed CACC algorithm. Like P1, the Flatbed algorithm maintains a constant vehicle spacing by using the CVS policy. This is achieved primarily by relying on local radar to measure the distance and speed of the preceding vehicle.

The Ploeg algorithm [23] employs a CTH control policy. This algorithm mainly relies on speed and acceleration data from the preceding vehicle. In the event of communication failures, the Ploeg algorithm allows vehicles to switch seamlessly between CACC and ACC via a custom hardware gateway. Hence, reliance on wireless communication networks is reduced.

Finally, Santini et al. [24] proposed a Consensus CACC algorithm that is based on the CTH control policy. This is implemented by getting position and speed information from all vehicles in a platoon. The consensus algorithm eliminates the need for a leader vehicle. Instead, each vehicle collects data from all other vehicles for decision-making.

2.3 Barrage jamming

Barrage jamming attacks, also known as noise jamming, are a major class of cyberattacks that can disrupt wireless communication either partially or completely. These attacks are relatively easy to execute, as the attacker continuously transmits noise-like energy across the entire frequency spectrum used by the communication channel [25].

One of the reasons for their effectiveness is that barrage jamming requires minimal knowledge of the target system beyond the basic protocol specifications, which are typically available in publicly accessible standardization documents. Relevant examples include IEEE 802.11p [26] and C-V2X [27]. The goal of this attack is to degrade the quality of the legitimate signal by lowering the signal-to-interference-plus-noise ratio (SINR).

According to Lichtman et al. [28], barrage jamming can be classified as time-uncorrelated and protocol-unaware. This means that barrage jamming is uncorrelated in time concerning the targeted signal and requires no detailed information about the communication protocol. Hence, it is easy to implement and execute. In contrast, nulling [29] or cancellation attacks [30] are time correlated and protocol aware, which makes it difficult for the attacker to design and implement the attacks. Researchers have developed and deployed these jamming attacks, which we discuss in this section.

Georgios et al. [31] evaluated the impact of barrage jamming attacks of varying intensities on vehicle communication, considering scenarios with both stationary and moving attackers. They proposed defense mechanisms, including beamforming, GPS, and onboard sensors, to enhance vehicle resilience against jamming attacks.

2.4 Jamming attacks on platoons

Several previous studies have investigated the consequences of jamming attacks on platooning systems. Hu et al. [32] studied the impact of jamming attacks on platoon stability. They used software-defined radios to demonstrate the feasibility of their attacks. Segata et al. [33] simulate jamming attacks using the Plexe simulation framework to demonstrate the effectiveness of fallback and recovery mechanisms. These mechanisms mitigate the impact of communication failures in cooperative driving applications. Van der Heijden et al. [6] proposed a general attack model to evaluate the resilience of three CACC algorithms implemented in Plexe. These CACCs are P1, Ploeg, and Consensus. Alipour-Fanid et al. [5] injected jamming attacks in an IEEE 802.11*p* communication protocol simulation model where the string stability and the safety of the CACC algorithms are evaluated.

What distinguishes our work from the studies discussed in this section is the way we model jamming attacks. We developed a realistic barrage jamming attack model, in which noise is injected at the physical layer of the communication system based on the IEEE 802.11p protocol. This approach allows us to capture both symmetric and asymmetric message losses, offering a more accurate representation of real-world communication losses.

Furthermore, to classify the outcomes, we base our analysis on collision incidents as well as the degree of vehicle deceleration. Previous studies have used other evaluation metrics such as distance variations or collision impact to classify the outcomes [5, 6]. Our work also contributes to validating the findings of these previous studies that evaluated similar CACC applications. Additionally, by applying a structured benchmarking framework, we ensure consistency and comparability of results.

3 Security benchmarking

In general, the primary motivation for defining benchmarks for computer-based systems is to provide a widely accepted and clear procedure for evaluating or comparing system implementations, components, or design solutions. When

it comes to basic concepts and main objectives, security benchmarking is akin to the closely related field of dependability benchmarking. In their work on dependability benchmarking, Kanoun et al. [34] identified "the main dimensions that are decisive for defining dependability benchmarks and the way experimentation can be conducted in practice". These dimensions are *(i)* the target system and benchmarking context, *(ii)* the measures to be evaluated, and *(iii)* the experimental conditions. We believe that these dimensions are also applicable to security benchmarking. Kanoun et al. also give examples of properties that a benchmark must achieve to be successful, such as *repeatability* (at least in statistical terms), *representativeness*, *portability*, and *cost-effectiveness*.

Since security benchmarking is a novel topic in the context of CDA systems, we would like to emphasize that our reference model for defining the security benchmarks is intended as an initial contribution towards how security benchmarks for assessing the jamming resilience of a CDA system could be defined.

3.1 Reference model for defining security benchmarks

This section describes our reference model for defining security benchmarks for CDA applications. The reference model's intended goal is to serve as a framework for discussing and developing security benchmarks based on attack simulation experiments. To this end, it assumes the availability of a simulation framework that makes it possible to study the impact of security attacks on road vehicles utilizing cooperative driving automation in various traffic environments.

We define a benchmark as a series of tests that we denote as a test campaign. Each test corresponds to exposing the System Under Test (SUT) to an attack at a specific time according to a given attack model. The benchmark is executed through attack injection, which refers to the process of deliberately subjecting the SUT to a specific attack.

Figure 1 illustrates the elements of the reference model and their interconnections. Some elements contain design information, such as data, knowledge, or specifications necessary to define, develop, and execute a system, model, or experiment. In contrast, other elements represent activities, such as setting up and executing the test campaign and analyzing the data. Elements like the SUT, driving scenario, and attack model have the design information. In contrast, campaign setup, benchmark execution, and data analysis involve activities that depend on information produced by other elements. Lastly, system response (raw data) and processed data elements store data.

The arrows indicate dependencies between elements, showing which components require information from other elements. Solid arrows represent primary relationships, where the receiving element requires data and knowledge from the source element. Dashed arrows denote secondary relationships, where the receiving element only requires knowledge of the source element.

We divide the benchmarking process into three phases: *campaign setup*, *benchmark execution*, and *data analysis*. The campaign setup defines a set of test cases, where each test is based on the knowledge and data about the SUT, traffic environment, driving scenario, and attack model.

The *SUT* setup involves several key activities that produce the information. These activities include *(i)* configuring the SUT size that is determining whether the benchmark

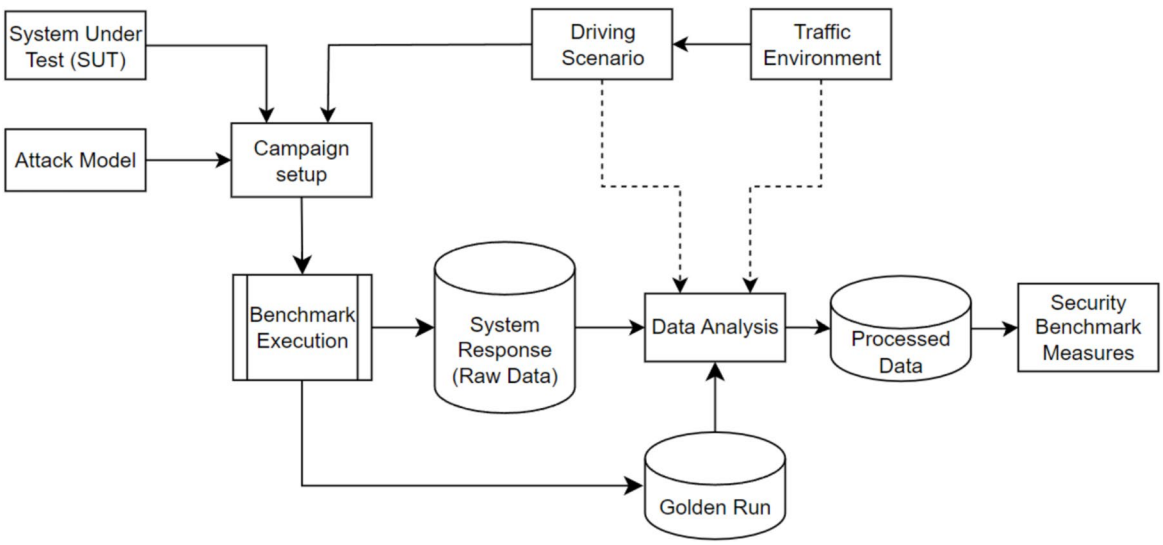


Fig. 1 A reference model for defining security benchmarks

applies to a single vehicle or multiple vehicles, *(ii)* selecting vehicle types such as cars, trucks, emergency vehicles, etc., *(iii)* specifying the control algorithm used by each vehicle such as ACC or CACC, *(iv)* setting control parameters such as engine characteristics, desired acceleration, and headway distance.

The *traffic environment* setup consists of *(i)* configuring road infrastructure, such as a single-lane or flat highway with or without elevation changes or frictional effects, *(ii)* including other traffic participants, such as oncoming traffic or additional vehicles, and *(iii)* setting weather conditions including fog, rain, snow, and sunlight.

The *driving scenario* setup includes *(i)* choosing a scenario type that specify the nature of the driving, and *(ii)* defining driving patterns, which include setting acceleration and deceleration profiles for vehicle(s).

The *attack model* represents the attack type used to evaluate the system's resilience. Common cyberattacks include barrage jamming, destructive interference (also known as nulling or cancelation attacks), and deceptive jamming [35], as well as spoofing and message falsification attacks [28].

Once all the reference model elements necessary for the campaign setup are configured, the activities of the ***campaign setup*** are carried out. These activities include: *(i)* selecting the attack target that is specifying whether the entire system or a subsystem is under attack, *(ii)* configuring attack parameters such as defining *attack start time*, *attack duration*, and *attack value*, and *(iii)* selecting the test execution method which include simulations with or without a graphical user interface (GUI).

The second phase of defining the security benchmark is ***benchmark execution***. This phase represents the process of executing the test campaign. Benchmark execution requires an attack injection tool (attack injector) capable of launching attacks and recording the system's response in a data repository. This tool facilitates systematic and repeatable testing, allowing system developers to assess vulnerabilities, analyze the impact of cyberattacks, and devise mitigation strategies to enhance the system's resilience against cyberattacks. This phase also includes golden run simulations and attack-free runs for baseline comparison. This ensures a reference point for evaluating the impact of cyberattacks on the system.

Data analysis is the third phase of defining the security benchmark. Data is first processed in this phase using evaluation metrics such as collisions and emergency braking. Based on these metrics, the experimental results are classified into different categories such as non-effective, negligible, benign, and severe. Note that the existing knowledge about the driving scenario and its intended environment is required for a complete and accurate data analysis.

Once the data analysis is completed and the *processed data* is logged, quantitative benchmark measures are obtained based on the evaluation metrics, e.g., number of collisions or near collisions. The measures are then used to evaluate the impact of security attacks on a CDA system. These measures assess the system's dependability, such as availability, accuracy, resilience, and performance, such as SPCf (SPECS in the presence of faults), THRf (throughput in the presence of faults), and RTMf(response time in the presence of faults) [36].

4 Security benchmarking of CACC algorithms - a case study

In this section, we present an example of a simulation-based security benchmarking approach to evaluate the jamming resilience of four different CACC algorithms. To perform this case study, we configured the experiments according to the three phases of the benchmarking process defined earlier: *campaign setup*, *benchmark execution*, and *data analysis*.

4.1 Campaign setup

To set up the test campaign, the SUT, traffic environment, driving scenario, and attack model must be determined. These elements are defined below.

4.1.1 System under test (SUT)

We select a platoon of four identical vehicles as our ***SUT***. Each vehicle in a platoon is equipped with a CACC algorithm to regulate speed and inter-vehicle distance. In our case study, the security benchmark evaluates four CACC algorithms: P1 [2] [8], Flatbed [13], Ploeg [14], and Consensus [24]. While P1 and Flatbed primarily rely on V2V communication, Ploeg and Consensus utilize onboard sensors. All algorithms are tested under ideal controller and engine parameters for consistency [8].

4.1.2 Traffic environment

For the ***traffic environment***, we consider a single-lane highway with no elevation changes or frictional effects. Additionally, no oncoming traffic or vehicles outside the platoon ensures an isolated testing scenario. The ideal weather conditions are considered, allowing us to focus on the impact of jamming attacks on the platooning application.

[2] This is the first CACC algorithm developed by Segata et al.

4.1.3 Sinusoidal driving scenario

Our security benchmark uses a ***sinusoidal driving scenario***, available in Plexe. This scenario simulates an extreme driving condition and is ideal for testing a platoon's string stability. In this scenario, we selected the vehicles to follow a sinusoidal driving pattern on a highway where they periodically accelerate and decelerate with a frequency of $0.2\ Hz$ and an amplitude of $5.0\ km/h$. The lead vehicle's maximum speed is set to $100\ km/h$, which all the trailing vehicles follow. The distance between the vehicles is set to 5 meters. The total simulation time for this scenario is set to $45\ s$.

We selected this driving scenario because it effectively captures dynamic highway driving, including acceleration and deceleration phases. To further develop the security benchmarking framework, exploring additional scenarios that reflect real-world traffic conditions would be valuable in the future.

4.1.4 Attack model

We selected a barrage jamming ***attack model*** to enable a simple and constructive exchange of benchmark definitions and results. We implement the model by manipulating the noise power parameter originally used for Signal to Interference & Noise Ratio (SINR) calculation for the received signal in the Veins simulation framework.

The noise power parameter is used to model the impact of various sources of *noise* that can affect the received signal, such as channel and thermal noise. To this end, we use three basic parameters for the attack model: *attack start-time*, *attack duration*, and *attack value*. The attack start time defines the time when the attack starts on the timeline of the driving scenario. The attack duration defines the duration of the active attack. The attack value defines the severity of the attack, i.e., the amount of noise injected into the communication channel.

Inline with some past work [12, 35], we selected 13 attack start times from $17.0\ s$ to $21.9\ s$ with a step size of $0.4\ s$ to configure the attacks. We conducted attack simulations with 11 different attack durations for each attack start time. We vary the attack value from 0.04 to 1×10^{-5} mW, in steps of 0.04×10^{-5} mW, resulting in 25 experiments. We selected this range of attack values as we conducted tests that showed little or no difference in the results obtained for noise signal power values above 1×10^{-5} mW. In total, we performed 3575 attack simulations for the sinusoidal scenario for each CACC algorithm that we evaluated.

Our attack model simultaneously affects the sending and receiving capabilities of all vehicles in a platoon. However, the effect of this noise differs among the vehicles due to variations in legitimate signal strength, which is influenced by the distance from each vehicle to the lead vehicle. As a result, the noise can lead to asymmetrical message losses. Asymmetric message losses refer to an unequal loss of messages across different vehicles. The distance between each vehicle and the platoon leader varies, leading to differences in how effectively the jamming noise disrupts communication.

4.2 Benchmark execution

For *benchmark execution*, we use ComFASE [7]. It is an open-source simulation-based fault and attack injection tool built on top of Veins, a network simulation framework [9]. The simulation frameworks that are utilized by the ComFASE simulation environment are *(i)* OMNeT++ *v. 5.6.2* [10], *(ii)* Veins *v. 5.1* to simulate the V2V communication [9], *(iii)* SUMO *v. 1.9.2* to design, simulate traffic, and study traffic behavior [11] and *(iv)* Plexe *v. 3.0a2* that enables realistic platooning application simulation [8].

ComFASE injects attacks on the IEEE 802.11p physical layer model in Veins. Moreover, various types of jamming attacks can be modeled in ComFASE, such as *delay attacks*, *denial-of-service (DoS) attacks* [7], *barrage jamming*, *deceptive jamming*, and *destructive interference* [35].

4.3 Data analysis

Finally, for *data analysis*, we use the raw data collected from the SUMO environment. This includes vehicle speed, acceleration, deceleration, and collision occurrences, which are used to classify the results and perform data analysis.

To analyze the impact of jamming attacks on vehicles, we focus on two critical parameters extracted from the raw data: *vehicle deceleration* and *collision incidents*. These parameters are essential for categorizing the outcomes and offer valuable insights into vehicle string stability and safety under the influence of jamming attacks. Deviations in the acceleration signal indicate potential compromises or degradation in control mechanisms, while collision incidents directly measure the severity of system failures caused by the attack.

4.3.1 Evaluation metrics

To classify the outcomes of the simulations, we define four outcome categories based on the vehicles' deceleration profiles and the occurrence of collision events. These four categories are: *Non-effective:* the deceleration profiles of the vehicles in an attack simulation run are identical to those observed in the golden (attack-free) run. *Negligible:* the recorded maximum deceleration of the vehicles in the

attack simulation run (RMD_{attack}) is less than or equal to the recorded maximum deceleration in the golden run (RMD_{golden}).

The, RMD_{golden} is set to be $1.53\ m/s^2$ for our target scenario. *Benign:* the RMD_{attack} is greater than the RMD_{golden} and less than or equal to the 'maximum comfortable braking' value that is set to $5.0\ m/s^2$. *Severe:* a collision has occurred or the RMD_{attack} is greater than the 'maximum comfortable braking'.

5 Experimental results

In this section, we present the results of the simulation runs obtained for the case study presented in Section 4. We conducted 3575 simulation runs per test campaign to evaluate each CACC algorithm. We divide the outcomes of the attack simulations into four categories: non-effective, negligible, benign, and severe (see Section 4.3.1). Severe outcomes represent simulations resulting in collisions or emergency braking, while the other categories represent no to less significant consequences. In this section, we focus the analysis on severe outcomes as they correspond to the system's lack of resilience.

The simulation results are presented by illustrating the impact of attack duration as well as start time across all simulations. Note that the classified results for each attack duration account for all attack start times and noise values. Likewise, the classified results for each attack start time include all attack durations and noise values.

5.1 P1 simulation results

5.1.1 Attack duration

In Fig. 3a, we see that for the P1 algorithm, the most prevalent outcome categories are severe outcomes (red bars) and benign outcomes (orange bars). Examining the severe outcomes, which account for 1475 (41.3%) out of total simulations, we observe an increasing trend as the attack duration extends. However, beyond 4 seconds, the number of severe outcomes stabilizes, showing minimal variation for each attack duration.

The second-largest category of outcomes is benign outcomes (orange bars), which includes cases where the deceleration values lie between $1.53\ m/s^2$ and $5.0\ m/s^2$. We observe an increasing trend for benign outcomes as the attack duration extends, reaching its peak at 3 seconds. Beyond this point, benign outcomes remain relatively stable across different attack durations.

Negligible outcomes (blue bars) account for 366 (10.2%) of the total outcomes. These outcomes are more prevalent for short attacks lasting less than 4 seconds. Non-effective outcomes (green bars) comprise approximately 143 (4%) of the total outcomes, and their distribution remains relatively consistent across all attack durations.

5.1.2 Attack start-time

We examine the impact of the attack start times on the outcome distributions, as illustrated in Fig 3b. The outcome presented in this figure strongly correlate to the acceleration and deceleration profiles of the driving scenario as shown in Fig. 2. The start times range from $17.0\ s$ to $21.8\ s$, where the intervals $17.0\ s$ to $17.4\ s$, and $20.2\ s$ to $21.8\ s$ represent acceleration periods, while the interval from $17.6\ s$ to $20.0\ s$ represents a deceleration period. Figure 3b shows that severe outcomes dominate when the attack start times coincide with an acceleration period, as indicated by the dark red curve.

In contrast, attacks initiated during the deceleration period predominately result in benign outcomes (orange curve), representing around 80% of the outcomes for the attacks initiated during this period. Furthermore, the negligible outcomes (blue curve) account for around 60% of the attacks initiated during the deceleration period.

Overall, less than 10% of the attacks result in non-effective outcomes (green curve). Interestingly, these outcomes are evenly distributed across all attack start times; hence, their percentage appears independent of the acceleration and deceleration periods.

The design of the P1 algorithm explains the vulnerability of the platooning system against jamming attacks during acceleration periods. When utilizing this controller, the lead vehicle periodically sends acceleration and deceleration commands to the trailing vehicles. If a vehicle loses communication, it will continue accelerating, decelerating, or keeping a constant speed according to the last received command. Consequently, if a jamming attack begins to block the communication channel during an acceleration period, the

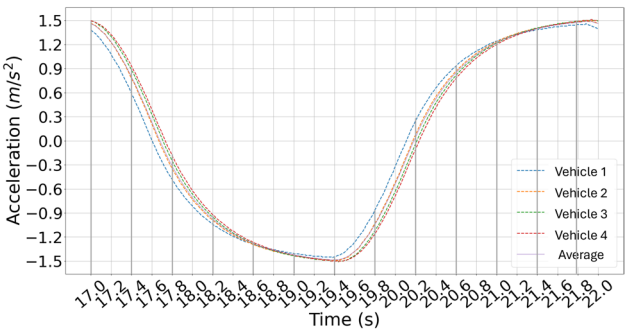
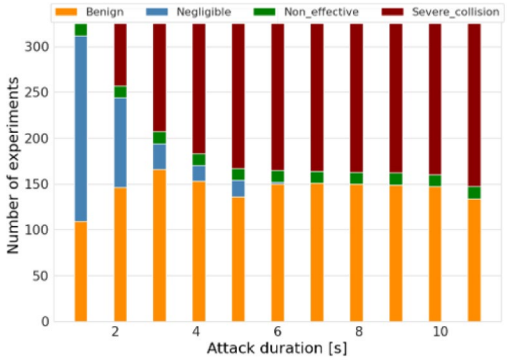
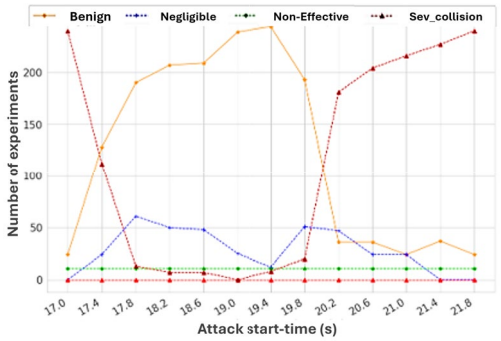


Fig. 2 Sinusoidal scenario: A sample simulation period showing acceleration profiles of a platoon

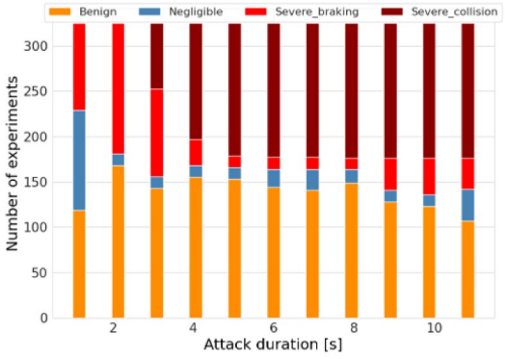


(a) P1:Outcome distribution w.r.t. the *attack duration*.

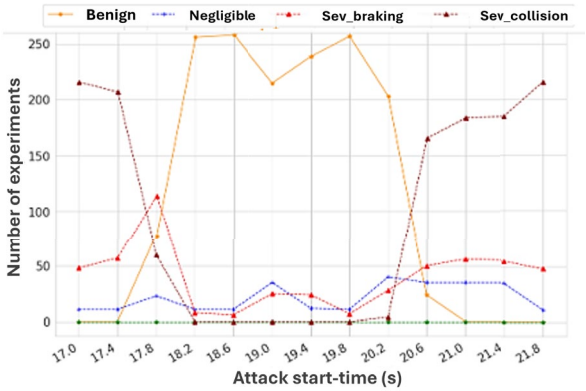


(b) P1:Outcome distribution w.r.t. the *attack start time*.

Fig. 3 Outcome distribution for jamming attacks on vehicles equipped with the P1 algorithm



(a) Flatbed:Outcome distribution w.r.t. the *attack duration*.



(b) Flatbed:Outcome distribution w.r.t. the *attack start time*.

Fig. 4 Outcome distribution for jamming attacks on vehicles equipped with the Flatbed algorithm

affected vehicles will continue to accelerate until the jamming attack ceases and the vehicles receive a deceleration or constant speed command from the lead vehicle.

5.2 Flatbed simulation results

5.2.1 Attack duration

Figure 4a illustrates the distribution of outcomes with respect to attack duration. The most prevalent categories evident in this figure are severe outcomes, which include severe braking (red bars), accounting for 526 (14.7%) of total outcomes, and severe collision (dark red bars), accounting for 1238 (34.6%). Given their impact, we have chosen to focus our analysis on these severe outcomes, as they are of utmost importance.

We observe that the severe collisions increase as the attack duration extends. The number of severe outcomes does not change much for each attack duration longer than 4 seconds. Notably, severe collisions dominate as the attack duration increases, indicating a higher risk of catastrophic failures for prolonged jamming attacks. This figure also shows that among severe outcomes, severe braking is dominant for shorter attack durations of less than 4 seconds. Benign outcomes account for 1530 (42.8%) outcomes where the system remains functional despite the attack. Negligible outcomes (blue bars), which total 281 (7.9%), are more significant for an attack duration of 1 second.

The Flatbed algorithm demonstrates slightly better resilience against barrage jamming attacks compared to the P1 algorithm. In the P1 algorithm, no severe braking is observed, meaning that when a vehicle loses communication, it does not attempt to brake to avoid a collision. In contrast, vehicles using the Flatbed algorithm attempt to mitigate severity when they come too close to each other. The algorithm successfully avoids collisions in only 14.7% of the total simulation runs.

5.2.2 Attack start-time

We now examine the impact of the attack start times on the outcome distributions, as illustrated in Fig. 4b. Similar to P1, the data depicted in this figure also correlate to

the acceleration and deceleration profiles of the driving scenario (see Fig. 2). Moreover, Fig. 4b shows that severe outcomes dominate when the attack start times coincide with an acceleration period, as indicated by the red curve. In contrast, attacks initiated during the deceleration period predominately result in benign outcomes (orange curve). The negligible outcomes (blue curve) are slightly more in the acceleration period.

5.3 Ploeg simulation results

Figure 5 shows the distribution of outcomes for attack duration. The observed outcomes include benign, negligible, and severe braking. The most common outcome is benign (represented by orange bars), accounting for 2840 simulations (79.4%). Negligible outcomes account for 711 simulations (19.9%), while severe braking was observed in just 9 simulations, all occurring for attack durations longer than 8 seconds.

The Ploeg algorithm outperforms both P1 and Flatbed. Despite a few severe braking incidents occurring at longer attack durations, it experiences no collisions across any attack duration. This superior performance is due to the algorithm's robust design, which makes it more resilient to jamming attacks. The Ploeg algorithm enables the vehicles to maintain safe inter-vehicle distances, even at high speeds. By utilizing wireless communication and fallback mechanisms, the algorithm enhances the safety of the vehicles and string stability of the platoon under attack.

We exclude the figure showing the results of Ploeg for attack start-times, as the majority of the outcome is not severe. We observe that the attack start time has minimal impact on the outcome, with only benign (79.5%) and negligible (19.8%) results. Only a few emergency braking events (0.25%) are observed when the attack duration exceeds 8 seconds. This outcome is closely tied to the design of the CACC algorithm. The Ploeg algorithm enhances resilience by utilizing onboard vehicle sensors and inter-vehicle communication to maintain platoon speed and string stability. Additionally, it incorporates a fallback mechanism to mitigate system failures. These features collectively make the Ploeg algorithm more resistant to jamming attacks.

5.4 Consensus simulation results

The consensus algorithm proved to be the best CACC algorithm among those evaluated using our security benchmarking approach, where all elements of the framework were kept constant. In Fig. 6, we observe no severe or negligible outcomes. Most outcomes are benign, accounting for 3551 (99.3%) simulation runs. We exclude the figure showing the results of Consesus for attack start-times, as the majority of the outcome are benign.

The jamming resilience of the consensus algorithm is due to the fact that it employs a distributed control strategy where each vehicle communicates with its neighbors to reach a consensus on the desired speed and inter-vehicle distance. This distributed approach makes the system less vulnerable to single-point failures caused by cyberattacks. The algorithm also allows reconfiguring the control topology based on the network status, enabling it to adapt to communication losses caused by jamming attacks. Additionally, this algorithm compensates for outdated information caused by network delays, mitigating the impact of jamming attacks.

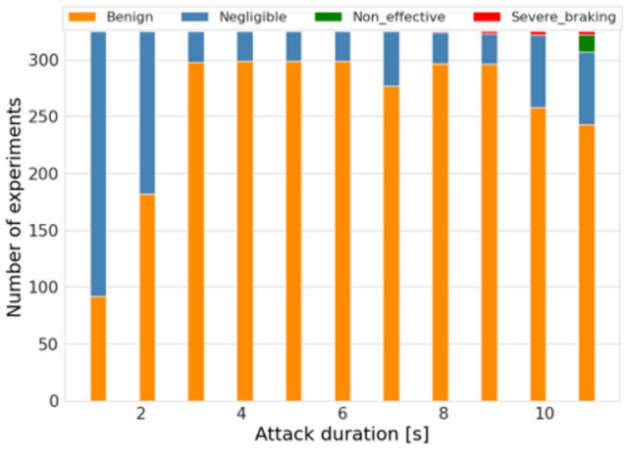


Fig. 5 Outcome distribution w.r.t. the *attack duration* where vehicles are equipped with the Ploeg algorithm

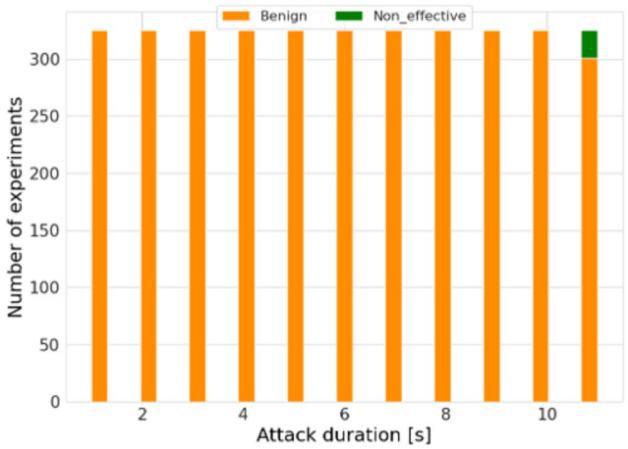


Fig. 6 Outcome distribution w.r.t. the *attack duration* where vehicles equipped with the Consensus algorithm

6 Discussions

6.1 Jamming resilience of CACC algorithms

Table 1 presents a summary of benchmark measures evaluating the jamming resilience of the four CACC algorithms with respect to the severe outcomes. The results indicate that the Consensus algorithm demonstrates the highest resilience against barrage jamming attacks, followed by Ploeg, which also exhibits strong resistance. In contrast, Flatbed and P1 are highly vulnerable to jamming attacks. These findings are consistent with those reported in previous studies [5–7, 35], which also highlight the resilience of CACC algorithms under varying communication failures.

The high resilience of the Consensus algorithm is its reliance on onboard sensors for maintaining inter-vehicle distance. Unlike other CACC algorithms, such as P1 and Flatbed, which depend solely on sensor data from the lead or preceding vehicles, the Consensus algorithm enables each vehicle to collect position and speed information from all other vehicles. This redundancy allows vehicles to verify and cross-check received data, making the system less vulnerable to isolated communication losses or corrupted messages.

The main limitation of the Consensus algorithm is that it reduces traffic density, as it requires larger inter-vehicle spacing to maintain safety when relying heavily on onboard sensors. These sensors could also be vulnerable to faults. In contrast, CACC algorithms that primarily rely on inter-vehicle communication can maintain shorter spacing and are more vulnerable to jamming attacks. However, they are less affected by individual sensor faults and errors due to their dependence on V2V communication data.

Overall, a CACC algorithm that combines V2V communication, onboard sensing, and robust fallback mechanisms offers improved control, safety, and security under jamming attacks.

6.2 Usefulness of our reference model

We propose a reference model for defining security benchmarks for CDA systems. This model offers a clear overview of the essential components and their interconnections, enabling a systematic approach to designing, executing, and analyzing tests to evaluate the impact of cyberattacks on the safety of CDA systems.

Our proposed reference model is designed to be flexible. It supports the detailed evaluation and analysis of a single CACC algorithm under various driving scenarios and cyberattacks, as demonstrated in our previous work [12]. In addition, it enables the comparison of multiple CACC implementations under consistent and repeatable testing conditions. This includes using the same driving scenario and attack model with identical attack parameters, as demonstrated in the current study.

Our reference model is flexible due to its modularity. Users can take advantage of the scenario-based testing methodology [37], can select specific attack types [28, 35] that meet their evaluation goals, apply appropriate evaluation metrics [5, 12] for the classification and analysis of results, and customize tests to assess the jamming resistance of the CACC algorithms in a repeatable and systematic way.

6.3 Threats to validity

6.3.1 Internal validity

Internal validity can be defined as the degree of confidence that the experimental design parameters are relevant and produce meaningful results.

In our benchmarking framework, the key experimental design parameters to evaluate a system under test include: *(i)* the test environment and its parameters, *(ii)* the driving scenario and its parameters, and *(iii)* the attack model and its parameters. Inaccurate or inappropriate choices of these parameters or their values would compromise the internal validity, making the results unreliable.

6.3.2 External validity

External validity can be defined as the extent to which results from an experimental study can be applied to other situations.

Table 1 Summary of the Benchmark Measures w.r.t Severe Outcomes

Scenario Type	Attack Type	Injection Mechanism	Target Parameter	SUT	Severe Outcomes
Sinusoidal Driving	Barrage Jamming	The jammer continuously emits noise energy across the entire frequency spectrum utilized for inter-vehicle communication [28]	Noise signal power [9]	P1	1475 (41.3%)
				Flatbed	1764 (49.3%)
				Ploeg	9 (0.3%)
				Consensus	0

The platooning application we evaluated involves four CACC algorithms, a sinusoidal scenario, a wireless channel model, and the barrage jamming attack model. Altering any of these potentially yields different outcomes. For example, employing other wireless channel models, such as two-way interference, or scenarios like constant-speed platooning, could yield different results.

7 Summary and future work

The security benchmarking of CDA systems is crucial for characterizing their behavior in the presence of cyberattacks. This work introduces a reference model for defining security benchmarks aimed at evaluating the jamming resilience of CDA systems. The model is structured into three main phases: *campaign setup*, *benchmark execution*, and *data analysis*.

To demonstrate the effectiveness of our proposed security benchmark, we conducted a case study using a *sinusoidal* driving scenario to evaluate the resilience of four different CACC algorithms (P1, Flatbed, Ploeg, and Consensus) to barrage jamming attacks. We analyze the impact of attack duration and attack start-time on the safety of these algorithms. A simulation-based test environment comprising Plexe, Veins, and SUMO simulation frameworks is utilized for evaluation.

The experimental results indicate that algorithms that rely primarily on external communication without fallback mechanisms, such as P1, are more vulnerable to jamming attacks. In contrast, algorithms that use vehicle sensors on board, external communication, and fallback mechanisms, such as Ploeg and Consensus, demonstrate greater resilience to barrage jamming attacks.

The experimental results reveal that attack start time influences the likelihood of collisions. In fact, barrage jamming is more effective when initiated during the acceleration period of a driving scenario. Moreover, the duration of the attack also plays a crucial role in increasing collision probability. However, beyond a certain threshold (approximately 4 seconds in our experiments), extending the attack duration does not lead to a significant rise in the number of severe outcomes.

As part of our **future work**, we plan to investigate the impact of varying *attack values* on the jamming resilience of CACC algorithms. The attack value is a key parameter of the jamming attack that determines the intensity of the jamming signal. It can significantly influence the safety and stability of CACC algorithm under attack. Moreover, as part of our future work, we intend to further validate our proposed benchmarking framework by evaluating CACC algorithms across a broader range of driving scenarios and traffic environments. In particular, we aim to incorporate a more realistic wireless channel model, such as *two-ray interference model* [38]. Another potential future direction is to implement a real-world barrage jamming attack to validate the simulation attack model.

Acknowledgments The work of this paper has been partly done in the context of the SUNRISE project, funded by the European Union's Horizon Europe Research and Innovation Actions under grant agreement no.101069573.

Author Contributions Mateen Malik authored the main manuscript. Behrooz Sangchoolie, as the industrial supervisor, and Johan Karlsson, as the main supervisor, oversaw the research. They provided critical feedback throughout the experimentation and writing process, ensuring clarity and coherence in both the result analysis and presentation.

Funding Open access funding provided by RISE Research Institutes of Sweden.

Data Availability No datasets were generated or analysed during the current study.

Declarations

Competing interests The authors declare no competing interests.

Open Access This article is licensed under a Creative Commons Attribution 4.0 International License, which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons licence, and indicate if changes were made. The images or other third party material in this article are included in the article's Creative Commons licence, unless indicated otherwise in a credit line to the material. If material is not included in the article's Creative Commons licence and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder. To view a copy of this licence, visit http://creativecommons.org/licenses/by/4.0/.

References

1. Committee CDAC, Taxonomy and Definitions for Terms Related to Cooperative Driving Automation for On-Road Motor Vehicles. SAE Standard J3216_202107. Accessed 29 Jul 2024. https://www.sae.org/standards/content/j3216_202107/
2. Lammert MP, Duran A, Diez J, Burton K, Nicholson A (2014) Effect of platooning on fuel consumption of class 8 vehicles over a range of speeds, following distances, and mass. SAE Intern J Commer Vehicles 7(2014-01-2438):626–639
3. Axelsson J (2016) Safety in vehicle platooning: A systematic literature review. IEEE Trans Intell Transp Syst 18(5):1033–1045
4. Petrillo A, Pescapè A, Santini S (2017) A collaborative control strategy for platoons of autonomous vehicles in the presence of message falsification attacks. In: 2017 5th IEEE International Conference on Models and Technologies for Intelligent Transportation Systems (MT-ITS), pp 110–115

5. Alipour-Fanid A, Dabaghchian M, Zeng K (2017) Platoon stability and safety analysis of cooperative adaptive cruise control under wireless rician fading channels and jamming attacks. https://doi.org/10.48550/arXiv.1710.08476
6. Heijden R, Lukaseder T, Kargl F (2017) Analyzing Attacks on Cooperative Adaptive Cruise Control (CACC). In: 2017 IEEE Vehicular Networking Conference (VNC), pp 45–52. https://doi.org/10.1109/VNC.2017.8275598
7. Malik M, Maleki M, Folkesson P, Sangchoolie B, Karlsson J (2022) ComFASE: A Tool for Evaluating the Effects of V2V Communication Faults and Attacks on Automated Vehicles. In: 2022 52nd Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN), pp 185–192. https://doi.org/10.1109/DSN53405.2022.00029
8. Segata M, Joerer S, Bloessl B, Sommer C, Dressler F, Cigno RL (2014) Plexe: A platooning extension for veins. In: 2014 IEEE Vehicular Networking Conference (VNC), pp 53–60. https://doi.org/10.1109/VNC.2014.7013309
9. Sommer C, German R, Dressler F (2011) Bidirectionally Coupled Network and Road Traffic Simulation for Improved IVC Analysis. IEEE Trans Mob Comput 10(1):3–15. https://doi.org/10.1109/TMC.2010.133
10. omnetpp.org: OMNet++ Simulation Models and Tools. https://omnetpp.org/. Accessed 27 Mar 2025
11. Lopez PA, Behrisch M, Bieker-Walz L, Erdmann J, Flötteröd Y-P, Hilbrich R, Lücken L, Rummel J, Wagner P, Wiessner E (2018) Microscopic Traffic Simulation using SUMO. In: 2018 21st International Conference on Intelligent Transportation Systems (ITSC), pp 2575–2582. https://doi.org/10.1109/ITSC.2018.8569938
12. Malik M, Sangchoolie B, Karlsson J (2024) A Simulation-based Security Benchmarking Approach for Assessing Cooperative Driving Automation (CDA) Applications. In: 2024 EAI International Conference on Intelligent Transport Systems (INTSYS). https://doi.org/Accepted_paper_for_INTSYS_Conference
13. Ali A, Garcia G, Martinet P (2015) The Flatbed Platoon Towing Model for Safe and Dense Platooning on Highways. IEEE Intell Transport Syst Mag 7(1):58–68. https://doi.org/10.1109/MITS.2014.2328670
14. Ploeg J, Semsar-Kazerooni E, Lijster G, Wouw N, Nijmeijer H (2015) Graceful Degradation of Cooperative Adaptive Cruise Control. IEEE Trans Intell Transp Syst 16(1):488–497. https://doi.org/10.1109/TITS.2014.2349498
15. Santini S, Salvi A, Valente AS, Pescapé A, Segata M, Lo Cigno R (2017) A Consensus-Based Approach for Platooning with Intervehicular Communications and Its Validation in Realistic Scenarios. IEEE Trans Veh Technol 66(3):1985–1999. https://doi.org/10.1109/TVT.2016.2585018
16. Conte TM, Hwu W-MW (1995) Advances in benchmarking techniques: New standards and quantitative metrics. In: Advances in Computers vol 41, pp 231–253
17. Oliveira RA, Raga MM, Laranjeiro N, Vieira M (2020) An approach for benchmarking the security of web service frameworks. Fut Gen Comput Syst 110:833–848. https://doi.org/10.1016/j.future.2019.10.027
18. Anisetti M, Ardagna CA, Damiani E, Gaudenzi F (2017) A Security Benchmark for OpenStack. In: 2017 IEEE 10th International Conference on Cloud Computing (CLOUD), pp 294–301. https://doi.org/10.1109/CLOUD.2017.45
19. Braun C (2007) NETCARBENCH: A benchmark for techniques and tools used in the design of automotive communication systems. In: IFAC Proceedings Volumes, vol 40, pp 321–328. https://doi.org/10.3182/20071107-3-FR-3907.00046
20. Shladover SE, Nowakowski C, Lu X-Y (2018) Using cooperative adaptive cruise control (cacc) to form high-performance vehicle streams. Definitions, literature review and operational concept alternatives
21. Kalogiannis K (2020) Investigating attacks on vehicular platooning and cooperative adaptive cruise control. PhD thesis, KTH
22. Rajamani R, Tan H-S, Law BK, Zhang W-B (2000) Demonstration of integrated longitudinal and lateral control for the operation of automated vehicles in platoons. IEEE Trans Control Syst Technol 8(4):695–708. https://doi.org/10.1109/87.852914
23. Ploeg J, Scheepers BTM, Nunen E, Wouw N, Nijmeijer H (2011) Design and experimental evaluation of cooperative adaptive cruise control. In: 2011 14th International IEEE Conference on Intelligent Transportation Systems (ITSC), pp 260–265. https://doi.org/10.1109/ITSC.2011.6082981
24. Santini S, Salvi A, Valente A, Pescapè A, Segata M, Lo Cigno R (2015). A Consensus-based Approach for Platooning with Inter-Vehicular Communications. https://doi.org/10.1109/INFOCOM.2015.7218490
25. Jirjees A (2017) Vehicular Ad Hoc Networks: Growth and Survey for Three Layers. Intern J Electric Comput Eng (IJECE) 7:271–284. https://doi.org/10.11591/ijece.v7i1.pp271-284
26. Jiang D, Delgrossi L (2008) IEEE 802.11p: Towards an International Standard for Wireless Access in Vehicular Environments. In: IEEE Vehicular Technology Conference, pp. 2036–2040. https://doi.org/10.1109/VETECS.2008.458
27. Generation Partnership Project (3GPP) (2022) Study on LTE support for V2X services. Technical report, 3GPP. http://www.3gpp.org/ftp/Specs/archive/22_series/22.185/22185-f00.zip
28. Lichtman M, Poston JD, Amuru S, Shahriar C, Clancy TC, Buehrer RM, Reed JH (2016) A communications jamming taxonomy. IEEE Secur Priv 14(1):47–54. https://doi.org/10.1109/MSP.2016.13
29. Mahal JA, Shahriar C, Clancy TC (2015) Emulated cp jamming and nulling attacks on sc-fdma and two novel countermeasures. In: MILCOM 2015 - 2015 IEEE Military Communications Conference, pp 275–280. https://doi.org/10.1109/MILCOM.2015.7357455
30. Moser D, Lenders V, Capkun S (2019) Digital radio signal cancellation attacks: an experimental evaluation. In: Proceedings of the 12th Conference on Security and Privacy in Wireless and Mobile Networks. WiSec '19, pp 23–33. Association for Computing Machinery, New York, NY, USA. https://doi.org/10.1145/3317549.3319720
31. Patounas G, Zhang Y, Gjessing S (2015) Evaluating defence schemes against jamming in vehicle platoon networks. In: 2015 IEEE 18th International Conference on Intelligent Transportation Systems, pp 2153–2158. https://doi.org/10.1109/ITSC.2015.348
32. Hu Y, Shan H, Dutta RG, Jin Y (2020) Protecting Platoons from Stealthy Jamming Attack. In: 2020 Asian Hardware Oriented Security and Trust Symposium (AsianHOST), pp 1–6. https://doi.org/10.1109/AsianHOST51057.2020.9358269
33. Segata M, Cigno RL, Hardes T, Heinovski J, Schettler M, Bloessl B, Sommer C, Dressler F (2023) Multi-Technology Cooperative Driving: An Analysis Based on PLEXE. IEEE Trans Mob Comput 22(8):4792–4806. https://doi.org/10.1109/TMC.2022.3154643
34. Kanoun K (2008). Dependability Benchmarking for Computer Systems. https://doi.org/10.1002/9780470370506
35. Maleki M, Malik M, Folkesson P, Sangchoolie B, Karlsson J (2022) Modeling and evaluating the effects of jamming attacks on connected automated road vehicles. In: 2022 IEEE 27th Pacific Rim International Symposium on Dependable

Computing (PRDC), pp 12–23. https://doi.org/10.1109/PRDC55274.2022.00016
36. Durães J, Vieira M, Madeira H (2004) Dependability Benchmarking of Web-Servers, pp 297–310. https://doi.org/10.1007/978-3-540-30138-7_25
37. Schuldt F, Ulbrich S, Menzel T, Reschka A, Maurer M (2015) Defining and Substantiating the Terms Scene, Situation, and Scenario for Automated Driving. In: Proceedings of the IEEE Intelligent Transportation Systems Conference (ITSC). https://doi.org/10.1109/ITSC.2015.164
38. Sommer C, Joerer S, Dressler F (2012) On the applicability of two-ray path loss models for vehicular network simulation. In: 2012 IEEE Vehicular Networking Conference (VNC), pp 64–69

Publisher's Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.