Assessing Safety of Adaptive Cruise Control Systems under CAN Denial-of-Service Attacks

Yucheng Ruan, Chengcheng Zhao, Zeyu Yang, Peng Cheng, Jiming Chen, Kasper Rasmussen

***Abstract*—Denial-of-service (DoS) attacks on the Controller Area Network (CAN) of a car can disrupt in-vehicle communication, posing a subtle yet critical threat to adaptive cruise control (ACC) systems whose operation is tightly coupled with real-time vehicle safety. While there is extensive prior work on CAN bus security and ACC safety, the safety impact of DoS attacks on closed-loop vehicle control under dynamic conditions remains insufficiently understood. In this paper, we present a systematic safety analysis of ACC closed-loop control under CAN bus DoS attacks. First, we introduce a virtual CAN bus with explicit arbitration on a closed-loop experimental testbed that integrates OpenPilot and CARLA, enabling a configurable CAN DoS message injector under diverse dynamic scenarios. Under CAN DoS attacks, closed-loop experiments reveal two distinct ACC responses: bounded regulation under near-steady lead-vehicle motion and transient safety degradation under time-varying lead-vehicle motion. To understand the former observation, we analyze the local stability of the OpenPilot-based ACC system under abstracted constant delay. The model-based analysis derives a theoretical upper bound on the admissible sensing delay that preserves asymptotic stability around the nominal car-following equilibrium. To assess the latter safety-critical transient behavior at runtime, we propose *SaferCAN*, a state-aware safety monitor that dynamically evaluates safety degradation using a reachable tube computation. Extensive experiments demonstrate that *SaferCAN* enables timely and reliable detection of safety-critical conditions under DoS attacks, achieving negligible false-positive rates and reducing DoS-induced alarm delays to approximately 50 ms across different driving scenarios and attack intensities.**

***Index Terms*—Controller Area Network, State Aware Safety Monitor, DoS Attack, ACC, ADAS**

I. Introduction

Adaptive cruise control (ACC) is a widely deployed driver-assistance feature that provides automated longitudinal vehicle control. It is now available on nearly all new vehicle models from major automotive manufacturers [1]. Recent market reports project that the global ACC system market will reach approximately USD 12.7 billion by 2034, underscoring its extensive and growing adoption in production vehicles [2]. At this scale, any systematic safety vulnerability in ACC could affect a large number of vehicles, elevating ACC safety from a feature-level concern to a system-level risk.

ACC is typically implemented as a distributed system composed of multiple sensors, actuators, and controllers. The operational safety of ACC thus fundamentally depends on how sensing, control, and actuation are realized and coordinated within the vehicle. Communication among these components is carried over an in-vehicle network, predominantly based on the Controller Area Network (CAN) bus, the *de facto* communication standard for automotive powertrain systems [3], [4]. Consequently, disruptions to CAN communication can directly degrade sensing in ACC, rendering vehicle safety inherently dependent on the availability of the in-vehicle network.

Prior research has shown that, as network connectivity in modern vehicles increases, the CAN bus can be accessed through remote or software-mediated entry points [5], [6]. As a result, Advanced Driver Assistance Systems (ADAS), such as ACC, which rely on timely and continuous CAN communication, are exposed to communication-layer attacks. Among the various attack vectors identified for the CAN bus, denial-of-service (DoS) attacks provide a particularly representative case for analyzing the impact of communication disruptions on control systems [7], [8]. Unlike attacks that manipulate specific signal semantics or proprietary message formats, DoS attacks primarily affect the timing and availability of CAN communication, and do not require detailed knowledge of message payloads [9]. Taken together, these characteristics motivate an investigation into how DoS-induced communication disruptions influence the behavior and safety performance of ACC.

Directly launching CAN DoS attacks on a real vehicle during ACC operation is unsafe and difficult to conduct under controlled and repeatable conditions. Existing studies approach this cyber–physical problem from different abstraction levels, as summarized in Table I. Specifically, CAN attack and intrusion detection system (IDS) studies primarily identify abnormal communication behavior from packet-level features and traffic statistics [10], [11]. These works are effective at detecting communication layer anomalies, but their outputs are rarely linked to the evolution of vehicles' physical states and further ACC safety analysis. When studying ACC safety analysis, researchers typically abstract the effects of DoS attacks as communication interruptions or frozen signals at the control-model level [12], [13], without capturing how the resulting disruption propagates through an executable ACC stack to affect vehicle behavior. As for studies that attempt to bridge

This work was supported in part by the Fundamental and Interdisciplinary Disciplines Breakthrough Plan of the Ministry of Education of China JYB2025XDXM103, in part by the National Natural Science Foundation of China under Grant 62273305, in part by the Zhejiang Provincial Natural Science Foundation of China under Grant LR25F030002, in part by Zhejiang University Special Project of the State Key Laboratory of Industrial Control Technology under Grant ICT2025C05, and in part by the China Scholarship Council (CSC).

Y. Ruan, C. Zhao, P. Cheng, and J. Chen are with the College of Control Science and Engineering, Zhejiang University, Hangzhou 310027, China (e-mail: yuchengruan@zju.edu.cn; chengchengzhao@zju.edu.cn; lunarheart@zju.edu.cn; cjm@zju.edu.cn).

Z. Yang is with the iTrust, Singapore University of Technology and Design, Singapore 487372 (e-mail: zeyu_yang@sutd.edu.sg).

Kasper Rasmussen is with the University of Oxford, OX1 2JD Oxford, U.K. (e-mail: kasper.rasmussen@cs.ox.ac.uk).

TABLE I: Comparison with related research lines.

Research Line	Representative Studies	Attack Model	Closed-loop ACC	Runtime Safety
CAN attack / IDS	TCE-IDS [10], CANival [11]	CAN DoS	–	–
ACC safety analysis	Qiu [12], Berdich [13]	Communication Disruption	✓	–
State-aware automotive defense	SAVIOR [14], SAID [15]	Message injection	–	✓
–	**This work**	**CAN DoS**	✓	✓

cyber anomalies and their physical consequences to improve automotive defenses, they validate potentially corrupted sensor data or injected control messages using vehicle dynamics and physical invariants [14], [15]. However, they abstract away the underlying CAN attack mechanism and primarily assess the consistency or immediate consequences of malicious inputs with respect to the ego vehicle's internal state.

To bridge this gap, we begin by establishing a closed-loop experimental testbed designed to capture ACC behavior under CAN DoS attacks. The testbed integrates OpenPilot [16], a production-grade advanced driver-assistance system deployed in real vehicles, with CARLA, a high-fidelity driving simulator [17]. Within this setup, we augment a virtual CAN bus that explicitly models CAN arbitration, whereby high-priority messages override the transmission of lower-priority messages. Building on this virtual bus, we further deploy a configurable CAN message injector that generates high-priority traffic, thereby emulating realistic DoS attacks under diverse and dynamic driving scenarios.

We conduct a set of exploratory experiments to observe how CAN DoS attacks affect the ACC system under different driving scenarios. The experiments reveal that the consequences of CAN DoS depend on how communication disruption interacts with the current car-following condition. To explain the bounded regulation observed when the lead vehicle travels at an approximately constant speed, we first develop a delayed closed-loop model for the OpenPilot-based ACC system under CAN DoS attacks. Then, we analyze the local stability of its regulation loop around a steady-state car-following equilibrium. To assess the transient safety risks under time-varying lead-vehicle motion, we introduce *SaferCAN*, a state-aware safety monitoring framework. *SaferCAN* operates in a passive monitoring mode by observing CAN traffic and decoding relevant vehicle signals. Based on short-term system identification and long-term disturbance characterization, *SaferCAN* constructs a lightweight runtime longitudinal dynamics model of the ego vehicle. Using this model, *SaferCAN* projects the future state evolution of the closed-loop system under DoS-induced control inputs over a finite horizon, evaluates the resulting reachable state tubes against safety-related criteria, and quantifies the worst-case degradation of safety margins. These assessments enable differentiated runtime responses to CAN DoS attacks on the CAN bus.

The contributions of this paper are summarized as follows:

- We develop a closed-loop testbed for controllable CAN DoS injection during ACC operation. Its virtual CAN bus emulates priority-based arbitration, reproducing contention-induced sensing starvation.
- We analyze the closed-loop impact of CAN DoS attacks on the OpenPilot-based ACC system. We explain its bounded regulation in near-steady car following through an OpenPilot-specific delayed closed-loop model, and assess its transient safety degradation using *SaferCAN*.
- We evaluate *SaferCAN* on 10,377 DoS-affected segments across diverse ACC driving scenarios and CAN DoS attack intensities, showing reliable and timely detection of safety-critical conditions with low false-positive rates.

II. Preliminaries

Here, we provide background on the CAN bus, CAN DoS attacks, and OpenPilot, which together establish the foundation for understanding the design of our simulation testbed and the subsequent system-level analysis.

A. CAN Bus

CAN bus is a lightweight, broadcast-based communication protocol designed to provide reliable real-time data exchange under strict timing constraints. Each CAN message is transmitted in the form of a data frame consisting of several fields, including the arbitration field (i.e., CAN ID) and the payload field. One key feature of the CAN protocol is its *ID-based priority arbitration mechanism*. Each message is assigned a unique CAN identifier, where lower numerical values correspond to higher transmission priority. When multiple ECUs attempt to transmit simultaneously, the bus arbitration process ensures that only the message with the highest priority is successfully transmitted, while other messages are delayed and require retransmission. Such a mechanism guarantees determinism for safety-critical signals, such as braking or longitudinal control commands.

B. Denial-of-Service (DoS) Attacks on the CAN Bus

CAN DoS attacks aim to degrade or completely suppress legitimate in-vehicle communication by exploiting the priority-based arbitration mechanism of the bus. The typical CAN DoS attacks continuously inject messages with very low CAN IDs, thereby monopolizing the CAN bus and preventing lower-priority messages from being transmitted [18].

Empirical studies on real vehicles have demonstrated that CAN DoS attacks can significantly distort message timing patterns and induce abnormal system behavior [9], even when the attack does not directly manipulate payload values. Importantly, the resulting impact on system behavior is highly dependent on the vehicle dynamics and operating conditions, making the safety consequences of CAN DoS attacks difficult

to characterize using communication-level metrics alone [19]. In this work, we focus on CAN DoS attacks as a representative and practically relevant threat model, and investigate how communication disruption propagates through the perception-planning-control pipeline, ultimately damaging vehicle safety.

C. OpenPilot Overview

OpenPilot is an open-source advanced driver assistance system developed by comma.ai [16]. It has been widely adopted as an experimental platform in recent studies on ADAS security and safety evaluation [20]. Specifically, OpenPilot provides Level-2 driver-assistance functions such as ACC and automated lane centering (ALC), supporting longitudinal and/or lateral vehicle motion while the human driver remains responsible for supervision [21]. In addition, OpenPilot can interface with the vehicle's CAN bus to read vehicle signals and issue control commands, while using its own forward-facing camera for perception. These properties make OpenPilot a suitable production-style platform for studying how CAN-level communication disruptions propagate through ACC operation.

In this paper, we use OpenPilot v0.8.9 and focus on its longitudinal control pathway. This version follows a clear perception–planning–control architecture, which facilitates the analysis of how the effects of delayed and lost CAN signals propagate through the control pipeline.

III. Simulating CAN DoS Attacks against ACC: Testbed and Initial Observations

To enable systematic analysis of CAN bus DoS attacks in a closed-loop control setting, we construct a simulation testbed that integrates a production-grade ACC stack with a realistic communication-layer attack model. The objective of this testbed is to reproduce the communication-layer behavior induced by DoS attacks on a real CAN bus, including priority-based arbitration contention, message delay, and frame starvation. Here, we present the design objectives of the platform, its architecture and arbitration model, the implementation of the DoS attack injector, and a validation study against real DoS CAN traces, followed by initial closed-loop observations.

A. Testbed Design Objectives and Abstraction Boundary

To support systematic investigation of CAN DoS attacks in a closed-loop control setting, the simulation testbed must accurately reflect how communication-layer disruptions propagate through the ACC stack and influence vehicle behavior. In particular, the platform should enable controlled injection of adversarial CAN frames while preserving the interaction between sensing, planning, and actuation in the control loop.

The primary design goal of the testbed is to construct a DoS attack injector whose effects emerge naturally from CAN arbitration dynamics rather than being artificially imposed by directly deleting or manipulating victim messages. Since CAN DoS attacks fundamentally rely on priority-based contention for bus access, faithful emulation of arbitration behavior is essential to ensure that message delay and starvation arise from the same mechanism as in real-world attacks.

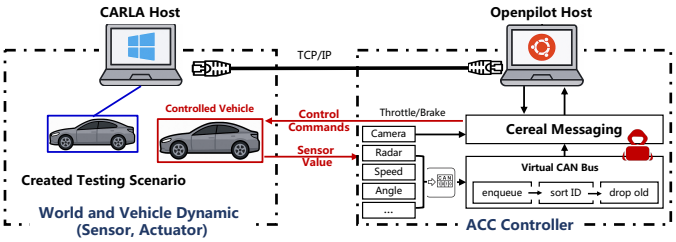


Fig. 1: Closed-loop co-simulation platform for CAN DoS attack experiments. The CARLA host provides the world and vehicle dynamics, while the OpenPilot host executes the ACC control stack.

Design Requirement. Accordingly, the testbed is required to satisfy the following requirements:

- Closed-loop execution fidelity. The platform must support execution of a representative ACC implementation without modifying its core control logic, ensuring that attack-induced communication distortions propagate through the actual perception–planning–control pipeline.
- Communication-layer realism. The CAN bus emulation must faithfully reproduce the arbitration process, contention resolution, and frame overwriting behavior.
- Configurable attack controllability. The system must enable control over attack parameters, including message priority, injection frequency, and duration, to allow systematic exploration of different attack intensities and operating conditions.

Hardware Abstraction Boundary. The testbed does not model hardware- and ECU-dependent behaviors, such as controller buffering, clock/jitter effects [22], communication-timeout handling [23], platform-specific gateway filtering [24], or hardware-induced latency. These mechanisms may mitigate, delay, or reshape the vehicle-level consequences of sensing starvation in a production ACC system. Therefore, the current testbed should be interpreted as a controllable closed-loop platform for studying CAN arbitration-induced sensing disruption, rather than as a complete hardware-in-the-loop validation of a production vehicle.

B. Closed-Loop Simulation

To enable controlled and repeatable evaluation of CAN DoS attacks in a closed-loop setting, we integrate OpenPilot as the ACC implementation with CARLA as the driving simulator. Since OpenPilot interfaces with sensors and actuators primarily through the CAN bus, it can be naturally coupled with CARLA by emulating CAN-based communication between the control stack and the simulated vehicle. The testbed further incorporates a configurable DoS injector, allowing controlled injection of adversarial traffic during closed-loop execution.

Overall architecture. The co-simulation testbed employed in this work is established on the basis of the framework proposed in [25], which integrates the CARLA v9.13 with the OpenPilot v0.8.9 to realize closed-loop ACC experiments. In comparison, as is shown in Figure 1, our new co-simulation platform

adopts a modular architecture that separates environment rendering from ACC computation, thereby improving execution efficiency and robustness under intensive DoS injection.

Specifically, CARLA, running on a Windows host equipped with a dedicated GPU, is responsible for real-time rendering and environment simulation. OpenPilot, deployed on a separate Ubuntu host, executes the autonomous driving control stack. Given the high data rate required for transmitting multi-modal sensor streams and real-time simulation states from CARLA, the communication load can reach the order of hundreds of megabits per second, motivating the use of a direct Gigabit Ethernet link to ensure reliable and low-latency data exchange. Overall, this decoupled architecture ensures reliable high-throughput and low-latency communication while providing a scalable and extensible foundation for closed-loop ACC experimentation under CAN DoS attacks.

Data flow and integration. Figure 1 illustrates the data flow of the close-loop platform. The CARLA host is responsible for generating the driving scenario and simulating vehicle dynamics, including both sensor outputs and actuator responses. At each simulation step, vehicle states and sensor measurements—such as radar observations, ego speed, steering angle, and other relevant signals—are produced by CARLA and transmitted to the OpenPilot host via a TCP/IP connection.

On the OpenPilot side, these sensor values are encoded into CAN-format messages and delivered through the virtual CAN bus layer before being consumed by the ACC control stack via the cereal messaging interface. The controller processes the incoming signals, performs perception–planning–control computations, and generates longitudinal control commands, including throttle and brake signals. These control commands are then transmitted back to the CARLA host, where they are applied to the simulated vehicle model, closing the loop.

C. CAN Bus Emulation with ID-Based Arbitration

Arbitration Model and Priority Handling. To enable fine-grained and controllable DoS simulation within the OpenPilot–CARLA framework, we introduce a custom virtual CAN bus that captures the priority-based access semantics of the CAN protocol. In prior implementations [25], sensor signals were directly published through the cereal messaging system to the `can` topic without arbitration, thereby neglecting contention effects inherent to real CAN communication.

In contrast, we introduce an additional arbitration layer that enforces CAN ID-based priority resolution. All outgoing CAN messages, including radar, speed, and other sensor signals, are first enqueued into a virtual bus queue. At each transmission cycle, message batches submitted by different senders are ordered according to their minimum CAN ID, where lower numerical IDs correspond to higher priority. Only the highest-priority batch is granted transmission access in that cycle. This abstraction preserves the key semantic property of CAN arbitration—namely, that higher-priority frames dominate bus access under contention—which is sufficient for modeling suppression effects induced by DoS attacks.

Overwrite Semantics and Message Suppression. In real CAN systems, controllers support automatic retransmission when arbitration is lost. However, periodic sensor messages are typically generated at fixed intervals and carry only the most recent measurements. Frames that repeatedly lose arbitration are often superseded by newer sensor updates rather than accumulating in a transmission queue [26]. Consequently, DoS attacks in practice manifest primarily as effective message suppression and data overwrite, instead of delayed burst transmissions after contention subsides. To reflect this behavior, our virtual bus discards lower-priority batches within each transmission cycle, intentionally modeling overwrite-driven suppression under heavy contention.

Algorithm 1 DoS Attack Runner

Initialize $dos\_active \leftarrow$ False, $dos\_end \leftarrow$ False
while not exit_event.is_set() **do**
  if $duration\_s = 0$ or dos_end **then**
    break
  end if
  if not dos_active and $frameIdx \geq T_{\text{trigger}}$ **then**
    $dos\_active \leftarrow$ **True**, record $start\_time$
  end if
  if dos_active **then**
    if $time - start\_time \leq T_{\text{duration}}$ **then**
      SEND_DOS_ATTACK(pm, $frameIdx$)
    else
      $dos\_active \leftarrow$ **False**, $dos\_end \leftarrow$ **True**
    end if
  end if
  sleep(T_{sleep})
end while

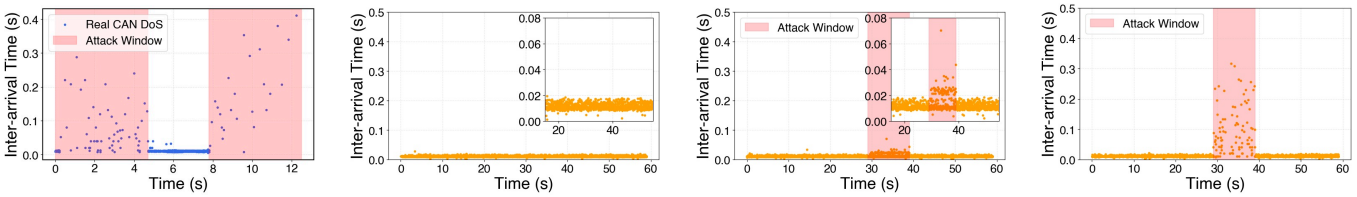
D. DoS Attack Injector Design

Design Principles. The objective of the DoS injector is to introduce controllable bus contention while preserving the closed-loop interaction between perception, planning, and control modules. Rather than modifying the internal logic of OpenPilot, the injector operates at the virtual CAN bus layer and competes for bus access under the same ID-based arbitration mechanism described in Section III-C.

The attack model follows three design principles. First, *priority dominance*: injected frames are assigned low numerical CAN IDs to ensure high arbitration priority under contention. Second, *temporal controllability*: the attack can be activated at a specified time and maintained for a configurable duration. Third, *intensity tunability*: the aggressiveness of the attack is controlled through the inter-injection interval, enabling systematic evaluation of different contention levels.

Injection Parameters and Algorithm. The DoS attack is governed by three parameters:

- T_{trigger}: the time at which the attack becomes active;
- T_{duration}: the duration of the attack window;
- T_{sleep}: a configurable inter-packet spacing parameter in the DoS injector implementation, which controls the relative injection frequency of malicious CAN messages during the attack window. Smaller values of T_{sleep} correspond to more aggressive DoS attacks.



(a) Real CAN DoS Attack. (b) Simulated No Attack. (c) Simulated Attack $T_{\text{sleep}} = 1e^{-2}$. (d) Simulated Attack $T_{\text{sleep}} = 1e^{-8}$.

Fig. 2: Comparison between real and simulated CAN DoS attacks. Subfigure (a) shows the inter-arrival time of message ID_130 under a real-world DoS attack [27], where the increased inter-arrival time indicates that the legitimate message is suppressed during the attack. Subfigures (b)-(d) display the inter-arrival time of speed signal message under simulated DoS attacks. The attack intensity is controlled by T_{sleep}, the sleep time of the malicious-message injection thread. The attack window is highlighted in red.

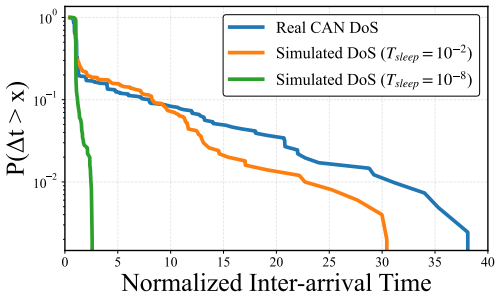
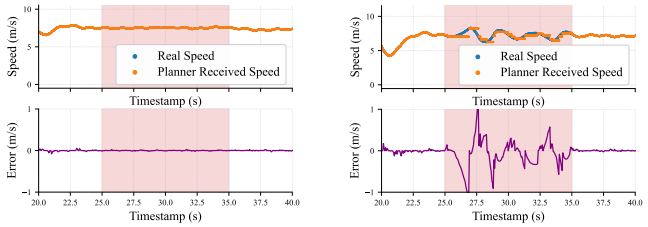


Fig. 3: Inter-arrival Time Distribution under Real and Simulated CAN DoS Attacks.



(a) Simulated Attack $T_{\text{sleep}} = 1e^{-2}$. (b) Simulated Attack $T_{\text{sleep}} = 1e^{-8}$.

Fig. 4: Impact of CAN Bus DoS Attacks on speed signals received by Longitudinal Planner of OpenPilot.

As is shown in Algorithm 1, we summarize the attack procedre of the dos attack injector. Specifically, the injector remains inactive until the trigger condition is satisfied. Once activated, it repeatedly injects high-priority CAN frames at intervals determined by T_{sleep}. The attack persists until the elapsed time exceeds T_{duration}, after which the injector deactivates.

The parameter T_{sleep} directly determines the attack intensity. Smaller values correspond to more frequent frame injections, resulting in increased bus contention and stronger suppression of lower-priority legitimate messages. Under the arbitration abstraction described earlier, injected frames with sufficiently high priority can consistently preempt competing traffic within each transmission cycle, thereby inducing message loss and effective sensing delays at the application layer. This design enables systematic exploration of attack severity by varying T_{sleep}, while maintaining reproducibility and precise temporal control within the closed-loop simulation framework.

E. Validation against Real CAN Traces Under DoS Attacks

Inter-arrival Time Pattern Comparison. Figure 3 compares real-world CAN DoS traces with our simulated attack results. Figures 2(a) shows the inter-arrival time of message ID_130 extracted from the HCRL dataset [27]. During the attack window (highlighted in red), the inter-arrival time increases, indicating sustained suppression and arbitration losses. Figure 2(b)–2(d) present the simulated inter-arrival patterns under different attack intensities controlled by the parameter T_{sleep}. In the no-attack case (Figure 2(b)), the inter-arrival time remains stable around the nominal period. When the attack is activated (Figures 2(c)–2(d)), the simulated traces exhibit a similar increase of inter-arrival time within the attack window. As T_{sleep} decreases, the density of suppressed intervals decreases, closely resembling the real trace shown in Figure 2(b). The qualitative similarity between real and simulated traces confirms that the proposed virtual arbitration and overwrite abstraction can faithfully reproduce DoS behavior.

Beyond the inter-arrival time increase, both real and simulated traces exhibit sustained starvation of lower-priority messages during the attack window. Rather than forming delayed bursts after contention subsides, suppressed messages are effectively overwritten by newer sensor updates. This starvation pattern aligns with practical CAN implementations, where periodic signals carry only the most recent state information.

Statistical Consistency. To further quantify similarity, we examine statistical characteristics of the inter-arrival distribution. Figure 3 presents the complementary cumulative distribution function (CCDF) of the normalized inter-arrival time of the victim CAN message under real and simulated DoS attacks. Each curve represents $P(\Delta t > x)$, where Δt denotes the inter-arrival time between consecutive successfully received instances of the victim ID, normalized by its nominal period. The CCDF provides a distribution-level view of suppression effects under bus contention.

The real CAN DoS trace exhibits a pronounced heavy-tailed distribution, indicating sustained starvation of the victim message during the attack window. As the attack intensity increases (i.e., as T_{sleep} decreases), the CCDF shifts toward larger

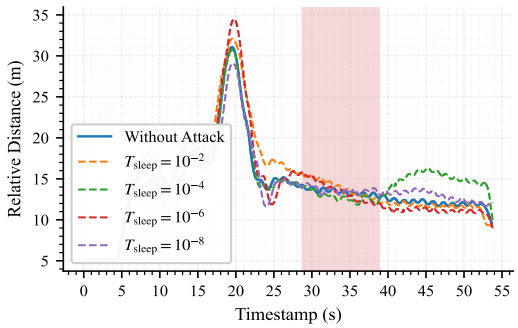


Fig. 5: Relative distance regulation of the ACC system under CAN DoS attacks with a steady lead vehicle. The shaded region denotes the attack window.

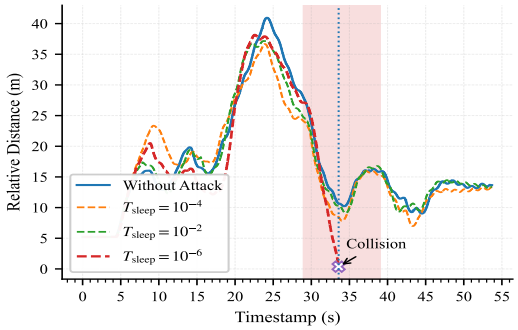


Fig. 6: Relative distance evolution under CAN DoS attacks with an unsteady lead vehicle. The main observation window is the shaded attack interval.

normalized inter-arrival values, reflecting longer suppression gaps and more severe arbitration losses. Importantly, the simulated curves closely align with the real trace across the entire distribution range, including both moderate delays and extreme tail events. This close agreement confirms that the proposed virtual arbitration mechanism faithfully reproduces not only the temporal suppression patterns but also the statistical structure of victim message timing under CAN DoS attacks. In particular, the similarity in tail behavior demonstrates that the simulation captures the dominant distributional signature of suppression-induced starvation.

F. Initial Closed-Loop Observations

To investigate the impact of CAN bus DoS attacks on ACC, we first examine their direct influence on the input signals to the longitudinal planner of OpenPilot. Figure 4 compares the real ego-vehicle speed (from CARLA) with the speed received by the planner under two simulated attack intensities. In the mild case $T_{\text{sleep}} = 1\text{e}^{-2}$ (Figure 4(a)), the received signal closely follows the ground truth, and the error remains negligible throughout the attack window. In contrast, in the severe case $T_{\text{sleep}} = 1\text{e}^{-8}$ (Figure 4(b), the attack introduces pronounced oscillations and spikes in the received speed, as shown in the error plot.

Then, we conduct preliminary experiments to see what would happen to the ACC system under DoS attacks. Specifically, we consider two representative lead-vehicle behaviors that correspond to distinct operating regimes of the ACC system. When the lead vehicle maintains a near-constant speed, the car-following task reduces to a regulation problem in which the ego vehicle attempts to maintain a fixed equilibrium defined by the desired spacing and matched velocity. In contrast, when the lead vehicle exhibits acceleration and deceleration, the ACC system operates under continuous external excitation. The following observations summarize the key phenomena in our experiments:

Observation I: As shown in Figure 5, when the lead vehicle maintains a near-constant speed, the ego vehicle can re-establish a bounded following distance after the transient disturbance across all tested attack intensities.

Observation II: Under an unsteady lead vehicle behavior, the same DoS attacks can lead to rapid distance collapse and collision. Figure 6 shows that stronger DoS attacks cause the ego vehicle to close in rapidly during the attack window, with the relative distance dropping to zero and resulting in a collision for $T_{\text{sleep}} = 10^{-6}$.

IV. Local Stability Analysis of the OpenPilot-based ACC under CAN DoS Attacks

In this section, we provide a model-based explanation for *Observation I*. Specifically, rather than relying on a generic ACC abstraction, we construct a delayed closed-loop model for the OpenPilot-based ACC system under CAN DoS attacks. Since the lead vehicle maintains an approximately constant speed (Figure 5), we examine the response of the OpenPilot regulation loop to DoS-induced delayed sensing without the continuous external excitation.

A. Vehicle Longitudinal Dynamics

To describe the longitudinal motion of the ego vehicle, we adopt a commonly used driving dynamics model derived from the principle of power balance:

$$m\dot{v} = -mg\sin\phi - \gamma mg\cos\phi - kv^2 + \frac{\eta}{R}T_{\text{en}}, \qquad (1)$$

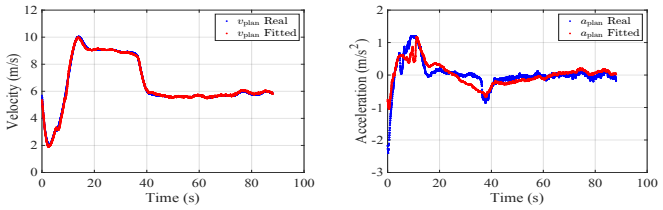
where m denotes the mass of the vehicle, ϕ denotes the road inclination angle, v denotes the longitudinal velocity, $\dot{v}$ denotes the vehicle acceleration, γ denotes the rolling resistance coefficient, g denotes the gravitational constant, k denotes the aerodynamic drag coefficient, η denotes the drivetrain efficiency, R denotes the wheel radius, and T_{en} denotes the engine torque.

B. Planning Module Approximation for OpenPilot

In OpenPilot v0.8.9, the planning module processes perception results to generate reference trajectories for the controller. Specifically, the longitudinal planner fuses information from radar and camera, including:

- Lead vehicle state: position, velocity, acceleration (from radar or model).
- Ego vehicle state: current speed, acceleration.

Based on this information, the planner constructs a desired speed and acceleration trajectory over a certain future horizon.



(a) Planned Velocity (b) Planned Acceleration

Fig. 7: Least-squares approximation of OpenPilot reference trajectories via linear range policy.

To model this planning logic, a corresponding desired velocity function $v_p(h, v)$ and desired acceleration function $a_p(h, v)$ can be constructed, mapping a spacing h and velocity v observation to a target speed and acceleration. Specifically, in this study, a linear regression model is employed to fit the range policy using empirical data. This modeling choice is motivated by the observation that, in car-following control, the planner's behavior can be well approximated locally by a linear range policy around typical operating points [28]. Accordingly, the fitted planner is formulated as $v_p(h, v)$ and $a_p(h, v)$. Specifically, in this study, a linear regression model is employed to fit the planner's implicit range policy using empirical data, formulated $v_p(h, v)$, $a_p(h, v)$ as:

$$v_p(h, v) = \alpha_1 h + \alpha_2 v + \alpha_3, a_p(h, v) = \beta_1 h + \beta_2 v + \beta_3, \quad (2)$$

where α_i, β_i are coefficients identified as below.

We conduct a fitting experiment using recorded trajectory planning data. We extract sequences of ego vehicle speed, relative distance, and the planner's output velocity $v_{\text{plan}}(t)$, over a fixed time window. To address the inherent delay between observation and planner output, we introduce a tunable offset parameter τ, aligning the inputs $(h(t-\tau), v(t-\tau))$ with the corresponding planner output $v_{\text{plan}}(t)$:

$$\min_{\alpha} \sum_{t} \left(v_{\text{plan}}(t) - (\alpha_1 h(t-\tau) + \alpha_2 v(t-\tau) + \alpha_3)\right)^2. \quad (3)$$

The optimal delay $\tau^\star$ is determined by minimizing the mean squared error (MSE) across all candidates. The fitted coefficients $\alpha = [\alpha_1, \alpha_2, \alpha_3]^\top$ corresponding to the best $\tau^\star$ yield a compact approximation of the implicit velocity policy. The resulting fitting curve is illustrated in Figure7, demonstrating a strong alignment between the predicted v_{plan}/a_{plan} and the actual values.

In this version of OpenPilot, the output of the planning module is passed to the longitudinal control module, which tracks the desired trajectory using a PID-based controller with feedforward acceleration compensation.

C. Longitudinal Controller of OpenPilot

To focus on the control-level impact of communication-layer attacks, we do not model the physical dynamics of the engine and actuator in detail. Instead, we assume ideal torque tracking, i.e., the engine torque T_{en} responds instantaneously to the control command generated by the OpenPilot's longitudinal controller:

$$\frac{\eta}{R} T_{\text{en}} = m\left(K_p \dot{z}(t) + K_i z(t) + K_f \left(a_p(h(t), v(t))\right)\right), \quad (4)$$

and

$$\dot{z}(t) = v_p\left(h(t), v(t)\right) - v(t). \quad (5)$$

Here, the gains K_p, K_i, and K_f correspond to proportional, integral, and acceleration feedback terms, respectively.

TABLE II: Data of a Tesla vehicle model from the CARLA simulator and the OpenPilot v0.8.9 stack are used for the parameters and system modeling in this study.

Symbol	Value	Unit	Description
k	0.46	kg/m	Air drag coefficient factor
m	2326	kg	Vehicle mass
K_p	0.77	1/s	Proportional gain
K_i	0.16	1/s^2	Integral gain
K_f	1.00	N·s/m	Feedforward gain

D. Car-Following Model for OpenPilot-based ACC system under DoS Attacks

In the presence of DoS attacks on the CAN bus, timely data transmission from sensors can be disrupted. This results in delayed availability of critical state measurements of inter-vehicle distance and relative speed for the planner module. To capture such signal degradation, we introduce a fixed sensing delay σ only into the control loop. Such an approximation has been widely adopted in the analysis of cruise control systems with communication imperfections, where time-varying or stochastic delays are replaced by their mean values to enable tractable stability analysis [28]. Following the equilibrium-linearization procedure used in analysis [28], we analyze the OpenPilot-based ACC system locally around a steady-state car-following equilibrium. Different from formulation in [28], the OpenPilot longitudinal planner provides both a desired velocity policy $v_p(h, v)$ and a desired acceleration policy $a_p(h, v)$; the latter enters the controller through the acceleration feedforward gain K_f. The car-following scenario can be modeled by the following system of differential equations with the lead vehicle velocity $v_L(t)$:

$$\begin{cases} \dot{h}(t) = v_L(t) - v(t), \\ \dot{v}(t) = -g\sin\phi - \gamma g\cos\phi - \dfrac{k}{m} v^2(t) + K_p\, \dot{z}(t-\sigma) \\ \qquad\quad + K_i\, z(t-\sigma) + K_f\, a_p\left(h(t-\sigma), v(t-\sigma)\right), \\ \dot{z}(t) = v_p\left(h(t), v(t)\right) - v(t). \end{cases} \quad (6)$$

With a constant-speed lead vehicle, the steady state (h^*, v^*, z^*) of the OpenPilot-based ACC system model above with a constant-speed lead vehicle $(v_L(t) = v_L^*)$ satisfies $\dot{h}(t) = 0, \quad \dot{v}(t) = 0, \quad \dot{z}(t) = 0.$

Let

$$x(t) = \begin{bmatrix} \tilde{h}(t) \\ \tilde{v}(t) \\ \tilde{z}(t) \end{bmatrix} = \begin{bmatrix} h(t) - h^* \\ v(t) - v^* \\ z(t) - z^* \end{bmatrix}, \quad u(t) = \tilde{v}_L(t) = v_L(t) - v^*,$$

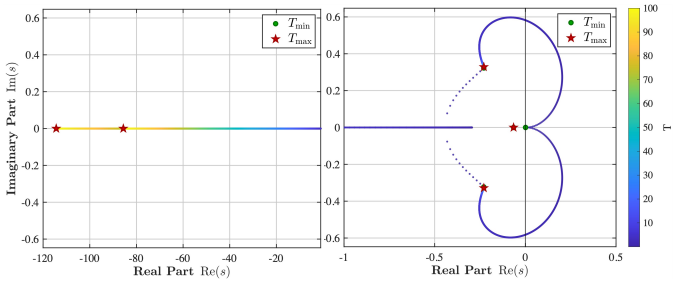


Fig. 8: Root locus of the system characteristic equation as the parameter T varies. The left subfigure illustrates the root trajectories on the real axis within a wide range, while the right subfigure shows a zoomed-in view near the imaginary axis. The color indicates the value of T, where the green circle ($T_{\min}$) and red stars ($T_{\max}$) denote the extreme cases.

the first-order expansion of OpenPilot-based ACC model around (h^*, v^*, z^*) be written as:

$$\dot{x}(t) = \mathbf{A}\, x(t) + \mathbf{A}_\sigma\, x(t-\sigma) + \mathbf{B}\, u(t), \tag{7}$$

with matrices

$$\mathbf{A} = \begin{bmatrix} 0 & -1 & 0 \\ 0 & -\frac{2k}{m}v^* & 0 \\ N_v^* & M_v^* - 1 & 0 \end{bmatrix}, \quad \mathbf{B} = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix},$$
$$\mathbf{A}_\sigma = \begin{bmatrix} 0 & 0 & 0 \\ N_v^* K_p + N_a^* K_f & (M_v^* - 1)K_p + M_a^* K_f & K_i \\ 0 & 0 & 0 \end{bmatrix}.$$

Here, the delayed matrix A_σ contains not only the local sensitivities of the desired velocity policy $v_p(h, v)$, but also those of the desired acceleration policy $a_p(h, v)$ introduced by OpenPilot's planner feedforward term. The symbol N_v^*, M_v^*, N_a^* and M_a^* represents the derivative of $v_p(h, v)$ and $a_p(h, v)$ at the equilibrium h^* and v^*:

$$N_v^* = \left.\frac{\partial v_p(h,v)}{\partial h}\right|_{h=h^*,\, v=v^*} \quad M_v^* = \left.\frac{\partial v_p(h,v)}{\partial v}\right|_{h=h^*,\, v=v^*}$$
$$N_a^* = \left.\frac{\partial a_p(h,v)}{\partial h}\right|_{h=h^*,\, v=v^*} \quad M_a^* = \left.\frac{\partial a_p(h,v)}{\partial v}\right|_{h=h^*,\, v=v^*}$$

The characteristic equation of the system in (7) is

$$\det\left(s\mathbf{I} - \mathbf{A} - \mathbf{A}_\sigma e^{-s\sigma}\right) = 0. \tag{8}$$

Substituting the expressions of $\mathbf{A}$ and $\mathbf{A}_\sigma$ derived above into the characteristic equation, we obtain the following delay-dependent characteristic function with $\sigma > 0$:

$$\Delta(s) = s^3 + \frac{2k}{m}v^* s^2 + \left((1 - M_v^*)K_p - M_a^* K_f\right) s^2 e^{-s\sigma} + \left(K_p N_v^* + (1 - M_v^*)K_i + N_a^* K_f\right) s e^{-s\sigma} + N_v^* K_i e^{-s\sigma}. \tag{9}$$

E. Stability Analysis of Time-Delayed LTI Systems

To initiate the stability analysis based on the delay-dependent characteristic function in (13), we begin by assigning numerical values to the system parameters. The values are obtained through linear range policy in Sec.IV-B with the result of: $N_v^* = 0.0639$, $M_v^* = 0.8420$, $N_a^* = 0.1122$ and $M_a^* = -0.3946$. Other parameters are adopted from the configuration settings of the CARLA and the OpenPilot, which are summarized in Table II.

To assess the stability of the delay-affected system based on equation (9), we substitute numerical values for the system parameters and analyze the resulting characteristic equation. In the absence of delay ($\sigma = 0$), the system reduces to a third-order polynomial whose roots lie strictly in the left half-plane, confirming asymptotic stability. When $\sigma > 0$, the characteristic equation becomes transcendental due to the exponential delay term. To determine the admissible delay margin, we apply a standard delay-system stability analysis method based on the D-subdivision approach [29].

Since equation (9) contains a single delay term, we follow the formulation in [30] by introducing an auxiliary characteristic equation through the transformation $e^{-s\sigma} = \left(\frac{1-Ts}{1+Ts}\right)^2$, $T > 0$, which preserves the imaginary-axis crossings of the original system. Using this transformation, the stability boundary is identified by locating purely imaginary roots of the auxiliary characteristic equation. As illustrated in Figure 8, the critical crossings occur at $s = j\omega_i$ with corresponding values T_i, from which the maximum admissible delay is computed as $\tau_{\max} \triangleq \min_i \left\{\frac{4}{\omega_i}\arctan(\omega_i T_i)\right\}$.

Substituting the identified values $\omega_i = 0.582$ and $T_i = 2.295$ yields a maximum admissible delay of $\tau_{\max} = 6.4\,\mathrm{s}$, ensuring asymptotic stability of the ACC system under steady-state operation. As illustrated in Figure 2(d), even under a simulated DoS attack with an extremely small sleeping parameter $T_{\text{sleep}} = 10^{-8}$, the resulting inter-arrival time reaches only about $0.5\mathrm{s}$. This value is consistent with observations from real-world DoS datasets (approximately $0.4\mathrm{s}$), and remains well below the theoretical admissible delay derived in Section IV. The result provides an affirmative answer to the question raised by *Observation I* within the steady-state fixed-delay abstraction considered here. Specifically, following-distance regulation observed under near-constant lead-vehicle speed is consistent with a delay-tolerance property of the nominal ACC loop, rather than being specific to a single attack execution or experimental trajectory.

V. Safety Degradation under DoS Attacks: A State-Aware Analysis

In this section, we address the concern raised by *Observation II*. Unlike the preceding equilibrium-based stability analysis, this setting involves time-varying lead-vehicle behavior and attack-dependent sensing starvation caused by CAN DoS attacks, which can produce intermittent and unpredictable delays in the information available to the ACC controller. To address this issue, we present *SaferCAN*, a state-aware safety monitoring framework that continuously evaluates safety risks by projecting the system's possible future states under bounded

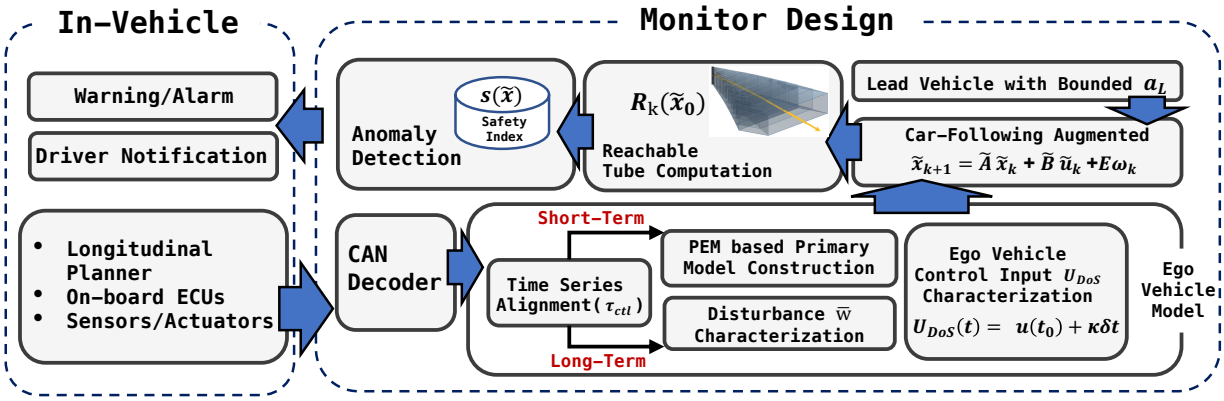


Fig. 9: The system architecture of *SaferCAN*, which passively observes CAN traffic and vehicle states, constructs a runtime model with bounded uncertainty, projects future reachable states, and issues tiered safety outputs.

uncertainty. As illustrated in Figure 9, *SaferCAN* passively observes CAN traffic and decodes it into relevant vehicle signals. Based on the aligned time series of the extracted signals, a lightweight runtime model of the ego-vehicle longitudinal dynamics, denoted as AlloM, is constructed using short-term system identification and long-term disturbance determination. By exploiting the observed behavior of the controller output under message suppression, *SaferCAN* characterizes the effective control input available to the ego vehicle during DoS attacks. By combining the lead-vehicle model with the components described above, *SaferCAN* formulates an augmented car-following state-space model and propagates the system states forward over a finite horizon. The resulting reachable sets are then evaluated against safety-related criteria to derive a scalar safety index, reflecting the worst-case degradation of safety margins over the prediction horizon. Finally, this safety index is mapped to tiered safety outputs, enabling differentiated responses including warn or remind.

A. Overview of Adaptive Linear Longitudinal Model

Vehicle longitudinal dynamics are inherently nonlinear. Detailed physics-based nonlinear models are often too complex for tractable analysis and real-time implementation. Prior data-driven investigations have demonstrated that a first-order linear longitudinal model is sufficient to capture the dominant input–output dynamics of real vehicles. While offering significant advantages in simplicity and real-time applicability, It can even outperform nonlinear physical models in terms of predictive accuracy [31]. Motivated by this, we employ the proposed adaptive linear longitudinal model (AlloM) with first order, providing an online, disturbance-aware linear over-approximation of the ego-vehicle's longitudinal dynamics. The primary model identified in this step takes the form:

$$\dot{x}_e(t) = A_e x_e(t) + B_e u_e(t), \qquad v(t) = C_e x_e(t), \tag{10}$$

with the inputs $u_e(t) = [u_{\mathrm{th}}(t),\, u_{\mathrm{br}}(t),\, u_{\mathrm{ri}}(t)]^\top$ consisting of throttle, brake, and road inclination, and with the ego-vehicle longitudinal velocity $v(t)$ as the output. Here, $x_e(t)$ denotes the state of the identified linear longitudinal model. It should be noted that this state does not necessarily correspond to explicit physical quantities, but provides a compact internal representation that captures the input–output dynamics of the ego vehicle [31]. The coefficient matrices A_e, B_e, and C_e are identified using the Prediction Error Method (PEM) based on measured input–output data over a receding time horizon. Thus, the resulting state-space model represents a short-term, local approximation of the longitudinal dynamics around the current operating condition.

Despite the accuracy of the identified linear model, prediction errors are inevitable due to unmodeled dynamics and disturbances. In safety-critical applications, these errors must be explicitly accounted for to balance conservativeness and informativeness. Accordingly, the PEM residuals are interpreted as the aggregated effect of unmodeled dynamics and mapped into an equivalent state perturbation, from which a bounded representation of the output uncertainty is constructed. Incorporating this additive uncertainty yields the disturbance-augmented model:

$$\dot{x}_e(t) = A_e x_e(t) + B_e u_e(t) + G_w w(t). \tag{11}$$

In practice, this formulation amounts to selecting an appropriate disturbance signal $w(t)$ that captures the effect of modeling error. The specific construction of $w(t)$ is detailed in the following subsection.

B. Online adaptation & deployment AlloM on the vehicle

In a real vehicle, the CAN bus continuously broadcasts a large number of sensor and controller messages, including signals such as wheel-speed, throttle/brake commands, etc. These informative real-time signals evolve at high frequency, reflecting time-varying vehicle dynamics. Thus, to tightly synchronize the estimated linear dynamic model with the real dynamics, we perform model identification in an online adaptive manner: model parameters are continuously re-estimated from recent CAN measurements, and the updated model is subsequently employed to compute short-horizon reachable sets of the ego-vehicle states.

Extraction of Signals as Aligned Time Series from CAN Bus Data. After obtaining traffic in each short sliding window of past CAN frames with length T_w (typically 3–4 s), we

extract the throttle [32], brake, road inclination [33], and velocity based on reverse engineering or DBC files. Due to actuation, communication, and processing delays on the CAN bus, control inputs (e.g., throttle and brake commands) do not affect the vehicle dynamics instantaneously. Therefore, a careful temporal alignment between control inputs and vehicle responses is required before system identification. To this end, we explicitly construct input–output pairs by mapping each control-command segment to a subsequent response interval of the ego vehicle. Specifically, a control input segment extracted over a window on the control-command time axis is associated with a delayed output segment on the response time axis, whose offset accounts for the estimated actuation and communication delay. This alignment ensures that each input segment corresponds to the vehicle response it causally influences, rather than to temporally coincident but dynamically unrelated measurements. By sliding this mapping along the CAN data stream, we obtain time-aligned sequences of control inputs and vehicle outputs, which are then used to form the training and validation sets for online parameter estimation and short-horizon prediction.

Primary Model Acquisition & Disturbance Characterization. Let T_w denote the short sliding window used for online parameter fitting and let T_{val} denote the validation interval for prediction. With sampling period T_s, we write $N_{\mathrm{tr}} = \lfloor T_w/T_s \rfloor$ and $N_{\mathrm{val}} = \lfloor T_{\mathrm{val}}/T_s \rfloor$ for the number of samples in the training and validation segments, respectively. We denote the measured input and output sequences as $u^{\mathrm{true}}(k)$ and $y^{\mathrm{true}}(k)$ for $k \in \{0, 1, \ldots, N_{\mathrm{tr}} + N_{\mathrm{val}}\}$. We collect the training and validation portions as:

$$\mathbf{u}_{\mathrm{tr}} = \{u^{\mathrm{true}}(k)\}_{k=0}^{N_{\mathrm{tr}}-1}, \qquad \mathbf{u}_{\mathrm{val}} = \{u^{\mathrm{true}}(k)\}_{k=N_{\mathrm{tr}}}^{N_{\mathrm{tr}}+N_{\mathrm{val}}-1},$$
$$\mathbf{y}_{\mathrm{tr}} = \{y^{\mathrm{true}}(k)\}_{k=0}^{N_{\mathrm{tr}}-1}, \qquad \mathbf{y}_{\mathrm{val}} = \{y^{\mathrm{true}}(k)\}_{k=N_{\mathrm{tr}}}^{N_{\mathrm{tr}}+N_{\mathrm{val}}-1}.$$

With the time-aligned input–output data, we identify the continuous-time linear model using the PEM, formulated as the nonlinear optimization problem

$$\min_{\theta} \sum_{k=0}^{N_{tr}-1} \|y^{\mathrm{true}}(k) - \hat{y}(k;\theta)\|^2, \tag{12}$$

where θ collects the parameters of (A_e, B_e, C_e) and $\hat{y}(k;\theta)$ denotes the simulated output of the candidate model. The optimization follows a damped Gauss–Newton scheme with adaptive regularization, and employs gradient-based fallback steps to ensure convergence on non-convex error surfaces. Stability constraints are imposed throughout the search. This procedure produces a locally optimal model fitted to the most recent CAN segment.

For deployment and computational purposes, the identified The continuous-time model is converted into an equivalent discrete-time representation via zero-order hold discretization, using the sampling period T_s consistent with the CAN data acquisition rate (typically $T_s = 0.01\mathrm{s}$). Considering the discrete form of the primary model with the discrete parameters $A_{e,d}, B_{e,d}, C_{e,d}$ and initial state x_0, the output of the validation set could be expressed as:

$$\hat{y}(k) = C_{e,d}A_{e,d}^{k}x_0 + \sum_{i=0}^{k-1} C_{e,d}A_{e,d}^{k-1-i}B_{e,d}u^{\mathrm{true}}(k),$$

where $k \in \{N_{\mathrm{tr}}, \ldots, N_{\mathrm{tr}} + N_{\mathrm{val}} - 1\}$ and the residual takes the form: $\delta(y_k) := y^{\mathrm{true}}(k) - \hat{y}(k)$. In particular, we posit that the validation residual can be attributed to a bounded disturbance sequence $\{w_i\}$ injected through the disturbance channel, which yields:

$$|\delta(y_k)| = \sum_{i=0}^{k-1} |C_{e,d}A_{e,d}^{k-1-i}\, w_i| \le \left(\sum_{j=0}^{k-1} |C_{e,d}A_{e,d}^{j}|\right) \bar{w},$$

i.e., the observed deviation at step k can be expressed as the propagated effect of past disturbances filtered by the system dynamics. To obtain a data-driven bound on the per-step disturbance magnitude, we invert the inequality above. Let $h_k := \sum_{j=0}^{k-1} \left|C_{e,d}A_{e,d}^{j}\right|$, which characterizes how a unit disturbance injected at each past step is amplified through the system dynamics and projected to the output at time k. To balance conservativeness and informativeness, rather than relying on the worst-case deviation, we adopt a percentile-based bound on the disturbance magnitude. Specifically, we select $\bar{w}$ such that a prescribed fraction $p \in (0, 1)$ of the observed output deviations are covered. Let $k = \{N_{\mathrm{tr}}, \ldots, N_{\mathrm{tr}} + N_{\mathrm{val}} - 1\}$ denote the validation index set. The disturbance bound is then defined as:

$$\bar{w}(p) = \max\left\{\frac{\mathrm{quantile}_p\left(|\delta y_k|\right)}{h_k}\right\}_k = \frac{\mathrm{quantile}_p\left(|\delta y_k|\right)}{h_{N_{tr}}}, \tag{13}$$

where $\mathrm{quantile}_\alpha(\cdot)$ denotes the p-th percentile. A larger p covers rarer residual errors and local nonlinear effects but leads to a more conservative reachable set, whereas a smaller p provides tighter bounds but may miss infrequent modeling errors. Based on this, the AlloM takes the final discrete form:

$$x_{k+1} = A_{e,d}x_k + B_{e,d}u_k + w_k, \quad y_k = C_{e,d}x_k \quad |w_k| \le \bar{w}(p). \tag{14}$$

Propagation of AlloM. Given the discrete-time model above, we compute the evolution of the system state under bounded disturbance by propagating interval enclosures of the state forward in time. The next-step reachable interval admits the conservative update as below:

$$x_{k+1} = A_{e,d}x_k \,\oplus\, B_{e,d}u_k \,\oplus\, W, \tag{15}$$

where x_k, u_k, and W denote the interval sets of the state, input, and disturbance, respectively, and $\oplus$ is the Minkowski sum. This interval-based propagation provides a computationally lightweight yet sound over-approximation of the reachable state set.

C. Characterization of PID Controller during DoS Attack

When the ACC is subjected to a DoS attack via the communication bus, the sensing pipeline experiences intermittent interruptions that prevent fresh measurements from reaching the planner on time. This results in a temporary freeze of the planner output, during which the reference signal remains

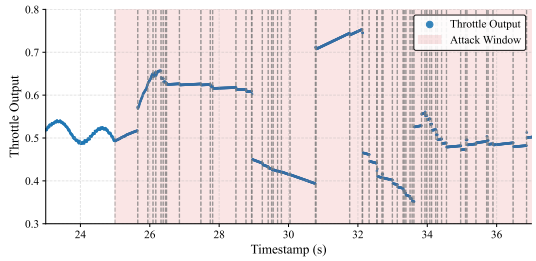


Fig. 10: Throttle output behavior under a DoS attack. During the attack window, vertical dashed lines indicate detected constant-output segments within the DoS interval.

equal to its last valid value. Recall that the planner output $z(t)$ appears in the equations (4)–(5). Under a DoS attack, the planner ceases to publish updates. During the attack period, we have $\dot{z}(t) = \dot{z}(t_0)$, i.e., $z(t)$ is effectively constant at the last published value $\dot{z}(t_0)$, resulting in the observation of linear increase/decrease during the communication blackout.

In general, the control output of the PID controller could be expressed as:

$$\boxed{U_{Dos}(t) = u(t_0) + \alpha \dot{z}(t_0)(t - t_0)}, \quad t \in [t_0, t_0 + \tau], \quad (16)$$

where α is a constant coefficient determined by the integral gain K_i together with the mapping from acceleration commands to throttle/brake inputs. $\dot{z}(t_0) = v_p(t_0) - v(t_0)$ denotes the difference of desired speed and the actual speed at the instant of the most recent planner update. Both $u(t_0)$ and $U_{\mathrm{Dos}}(t)$ denote the unified longitudinal command, obtained by merging the throttle and brake signals as follows:

$$u(t) = u^{\mathrm{th}}(t) - u^{\mathrm{br}}(t).$$

The DoS attack law in (16) is applied to obtain $U_{\mathrm{Dos}}(t)$, and then this unified command is split back into actuator signals:

$$u^{\mathrm{th}}_{\mathrm{Dos}}(t) = \max\{U_{\mathrm{Dos}}(t),\, 0\}, \quad u^{\mathrm{br}}_{\mathrm{Dos}}(t) = \max\{-U_{\mathrm{Dos}}(t),\, 0\}.$$

In this way, both $u(t_0)$ and $U_{\mathrm{Dos}}(t)$ undergo the same merge–split procedure when being mapped to the actual throttle and brake commands. Figure 10 illustrates this behavior for a representative DoS experiment with a sleep parameter of $T_{\mathrm{sleep}} = 10^{-6}$. As shown, once the planner output freezes, the PID controller produces a throttle command that drifts approximately linearly over the duration of the blackout, consistent with the prediction given by (16).

The control-input characterization in this subsection is specific to the OpenPilot-based ACC implementation used in our testbed, which is shown in Figure 10. *SaferCAN* uses this characterized control-input evolution as the attack-induced input model for reachable-tube propagation. For another ACC implementation, this characterization step must be repeated rather than directly reused. Different ACC systems may adopt different longitudinal-control policies, such as PID-, LQR-, or MPC-based controllers [34], and may obtain lead-vehicle information through different sensing architectures, such as radar, LiDAR, camera, or sensor-fusion pipelines [35]. These implementation choices determine which lead-state signals can become stale under DoS and how the controller converts such stale information into throttle or braking commands. Therefore, applying *SaferCAN* to another ACC system requires identifying the attacked sensing-to-control path and establishing a controller-specific DoS-induced control-input model. This model can be obtained experimentally from closed-loop DoS-injection tests or analytically when the controller structure and parameters are available.

D. Car-Following with Bounded Lead-Vehicle Behavior

Unlike the steady-state analysis performed earlier, we allow the lead vehicle to exhibit non-constant motion and model it as a normal driven vehicle. Accordingly, its acceleration is assumed to be bounded, reflecting a realistic driving scenario rather than an adversarial one [36]. Under this assumption, the control input follows:

$$\widetilde{u}_k = \begin{bmatrix} a_{L,k} \\ u_{e,k} \end{bmatrix}, \qquad a_L(t) = \dot{v}_L(t) \in [a_{L,\min},\ a_{L,\max}]. \quad (17)$$

Let $h(t)$ denote the spacing between the ego and lead vehicles, and $v_L(t)$ the velocity of the lead vehicle. The spacing evolution obeys the kinematic relation Furthermore, combining the ego-vehicle longitudinal model in (14), the complete car-following system can be written in the following augmented discrete-time state-space form:

$$\widetilde{x}_{k+1} = \widetilde{A}\,\widetilde{x}_k + \widetilde{B}\,\widetilde{u}_k + \widetilde{G_w} w_k, \quad (18)$$

where $\widetilde{G_w} = [0\ 0\ G_w]^\top$ and the augmented state $\tilde{x}_k$ for the car-following scenario, system matrices $\tilde{A}$, $\tilde{B}$ are given by:

$$\widetilde{x}_k = \begin{bmatrix} h_k \\ v_{L,k} \\ x_{e,k} \end{bmatrix}, \widetilde{A} = \begin{bmatrix} 1 & \Delta & -\Delta C_{e,d} \\ 0 & 1 & 0 \\ 0 & 0 & A_{e,d} \end{bmatrix}, \widetilde{B} = \begin{bmatrix} 0 & 0 \\ \Delta & 0 \\ 0 & B_{e,d} \end{bmatrix}.$$

We denote the admissible input and disturbance sets as $\mathcal{U}$ and $\mathcal{W}$, respectively. For a fixed initial condition $\tilde{x}_0$, the forward reachable set at step k is defined as the set of all states that can be generated by input/disturbance sequence:

$$\mathcal{R}_k(\tilde{x}_0) = \left\{ \tilde{x}_k \,\middle|\, \begin{array}{l} \exists\, \tilde{u}_0, \ldots, \tilde{u}_{k-1} \in \mathcal{U},\ \exists\, w_0, \ldots, w_{k-1} \in \mathcal{W} : \\ \tilde{x}_{i+1} = \tilde{A}\tilde{x}_i + \tilde{B}\tilde{u}_i + \widetilde{G_w} w_i, \quad 0 \le i < k \end{array} \right\}.$$

That is, $\mathcal{R}_k(\tilde{x}_0)$ is constructed as a conservative reachable set: it collects all states that can arise from any admissible sequence of inputs and disturbances over the prediction horizon, under the discrete-time model above. Serving as an outer approximation of all future trajectories that are compatible with the current condition and the disturbance bounds, $\mathcal{R}_k(\tilde{x}_0)$ is crucial for the subsequent safety monitor.

Overall, as illustrated in Figure 11, *SaferCAN* operates in an online manner after deployment on the vehicle and adopts a receding-horizon mechanism for runtime safety monitoring. *SaferCAN* neither updates the controller nor triggers message transmissions. Instead, it periodically recasts safety-threshold checking as a reachable-set monitoring problem under DoS-induced sensing starvation and produces only monitoring-level reminders or warnings. Specifically, at each runtime cycle, *SaferCAN* uses a short history window to update AlloM with

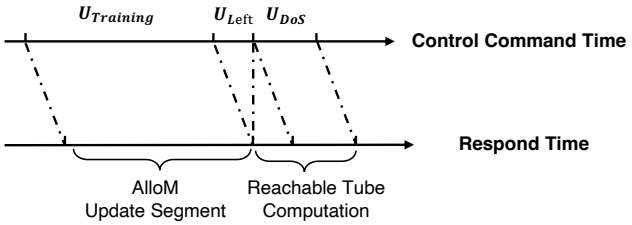


Fig. 11: Online receding-horizon operation of *SaferCAN*.

parameter $(A_{e,d}, B_{e,d}, C_{e,d})$. Given the updated model and the current measurement of system states $\tilde{x}_0$, SaferCAN computes a reachable tube $\mathcal{R}_k(\tilde{x}_0)$ over a fixed prediction horizon. The computation requires three ingredients: the augmented discrete car-following system according to equation (18), the initial state $\tilde{x}_0$, and control inputs. Concretely, the control inputs consist of two segments: (i) U_{Left}, the sequence of commands that have been issued by the controller but have not yet taken effect on the plant due to the delay described in Section V-B, and (ii) U_{DoS}, the sequence of inputs caused by DoS attack, according to equation (16). The reachable tube $\mathcal{R}_k(\tilde{x}_0)$ thus characterizes the worst-case evolution of system outputs from state $\tilde{x}_0$ over the prediction horizon under control inputs of U_{Left} and U_{DoS}.

E. Reachable Set based Safety Index Design

The reachable tube $\mathcal{R}_k(\tilde{x}_0)$ provides a set-based characterization of all possible system state evolutions under the considered control input uncertainty. Building upon this reachable set, we next design safety indices that explicitly account for the worst-case behaviors of safety-relevant variables over the prediction horizon. Generally, *SaferCAN* does not restrict safety assessment to a specific metric. Instead, safety requirements that depend on the system state can be expressed in terms of state-related safety variables, such as time-to-collision (TTC) [37], inter-vehicle distance margins [38], or other application-specific safety indicators. These requirements can be represented by scalar functions that map the augmented system state to quantities of interest for safety evaluation.

Formally, let $s(\tilde{x})$ denote a safety-related function defined over the augmented state $\tilde{x} \in \mathcal{R}_k(\tilde{x}_0)$. By propagating the reachable sets of the system states over a finite horizon, *SaferCAN* evaluates the values of $s(\tilde{x})$ within each predicted reachable set. This enables the monitor to conservatively assess the worst-case evolution of the selected safety variables over the prediction horizon, without relying on point estimates or instantaneous measurements.

Beyond evaluating a single worst-case safety value, *SaferCAN* further quantifies how likely the predicted system evolution is to enter unsafe conditions by measuring the coverage of unsafe states within the reachable tube. Based on safety index $s(\tilde{x})$, we define the unsafe state set as:

$$\mathcal{S}_{\text{unsafe}} \triangleq \{\, \tilde{x} \mid s(\tilde{x}) < 0 \,\}, \tag{19}$$

which represents states that violate the specified safety requirement. Let $\mathcal{R}_k$ denote the reachable set of the augmented system at prediction step k, obtained via forward propagation under bounded uncertainty. For each step k, we define the unsafe coverage ratio as

$$r_k = \frac{\text{vol}(\mathcal{R}_k \cap \mathcal{S}_{\text{unsafe}})}{\text{vol}(\mathcal{R}_k)}, \tag{20}$$

which quantifies the fraction of admissible future states that may enter the unsafe region at step k. Using the coverage ratio, *SaferCAN* adopts a tiered risk interpretation strategy over the prediction horizon. Specifically, let $r_{\max} = \max\{r_k\}$ denotes the maximum unsafe coverage over the finite horizon, the runtime safety response is then determined according to predefined thresholds:

- **Remind:** if $r_{\max} > r_{\text{remind}}$, indicating a substantial portion of admissible futures approach unsafe conditions;
- **Warn:** if $r_{\max} \geq r_{\text{warn}}$, indicating that unsafe behavior dominates the reachable set.

The prediction index k corresponds to a future time $t+k\Delta t$, where Δt denotes the prediction sampling interval. Therefore, the proposed monitor can be interpreted as a delay-aware risk detector over the prediction horizon. In particular, for a given risk threshold $r_{\text{th}} \in \{r_{\text{warn}}, r_{\text{remind}}\}$, let $k_{\text{th}} \in \{k_{\text{warn}}, k_{\text{remind}}\}$ denote the *earliest* prediction step at which the corresponding threshold is exceeded. The quantity $k_{\text{th}}\Delta t$ thus represents the *maximum admissible time delay* that the system can tolerate, under the current situation.

F. Computational Complexity and Potential Deployment

We further discuss the computational complexity of *SaferCAN* to clarify its real-time feasibility. *SaferCAN* consists of two computational parts: (i) the AlloM model update based on recent CAN measurements, and (ii) the reachable-tube propagation and safety-index evaluation.

Let N_w denote the number of samples in the sliding identification window, p the number of parameters in the identified longitudinal model, and I the number of iterations used by the PEM optimizer. At each iteration, evaluating the residual sensitivities and forming the associated least-squares terms over the identification window requires $O(N_w p^2)$ operations, while solving the resulting p-dimensional parameter-update problem requires $O(p^3)$ operations. Therefore, the overall complexity of the PEM-based AlloM update is approximately $O(IN_w p^2 + Ip^3)$. In our implementation, the ego longitudinal model is first-order, and the number of parameters is small.

For the runtime reachable-tube computation, let H denote the prediction horizon, n_a the dimension of the augmented car-following state, and m_a the input dimension. Since *SaferCAN* uses a discrete-time linear model with interval over-approximation, each propagation step consists of a dense state-set multiplication and an input-set multiplication, with costs $O(n_a^2)$ and $O(n_a m_a)$, respectively. Hence, the complexity of each propagation step is $O(n_a^2 + n_a m_a)$. The safety index $s(\tilde{x}) = \ell^\top \tilde{x}$ is linear in the augmented state, so evaluating it over the tube adds $O(Hn_a)$. Therefore, the online monitoring part of *SaferCAN* is polynomial in the state dimension and scales linearly with the prediction horizon. Additionally, with the purpose of CAN-based safety monitoring, the natural

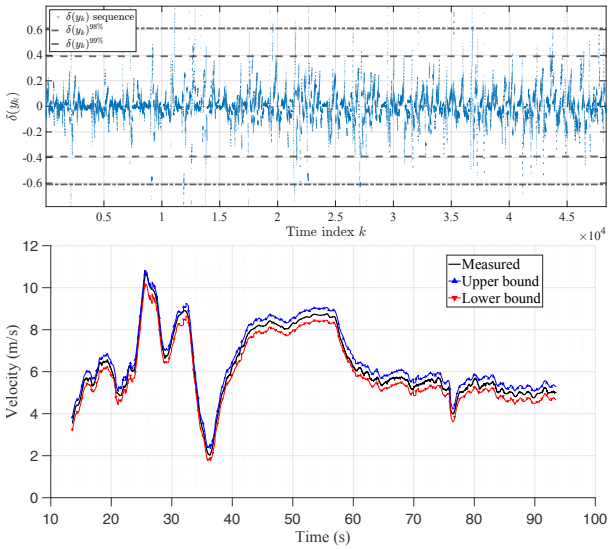
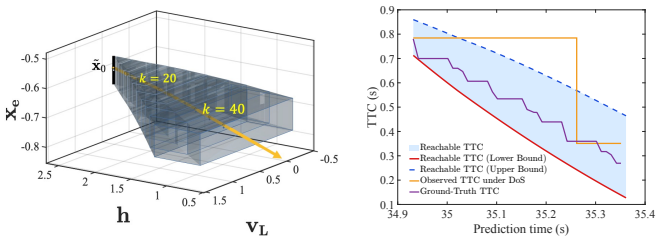
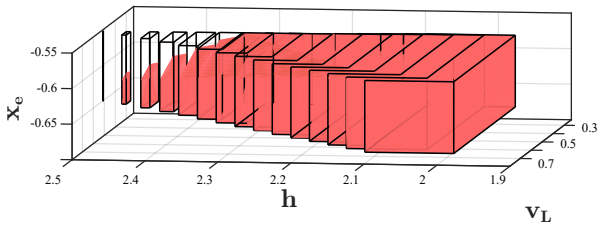


Fig. 12: Disturbance bound selection using empirical percentiles and validation of reachable output bounds. The 98% percentile of the output mismatch $\delta(y_k)$ is used to define the disturbance magnitude $\bar{\omega}$, resulting in predicted velocity bounds that fully enclose the measured trajectory.



(a) Reachable tube of the augmented car-following system.

(b) Reachable bounds of TTC.



(c) Safety interpretation based on TTC reachability.

Fig. 13: Safety Interpretation via reachable tube of TTC by *SaferCAN* under DoS Attacks.

deployment target for *SaferCAN* is a vehicle gateway or ADAS/domain controller. Modern automotive gateway and domain-controller platforms provide multi-core application processors and real-time cores with native support for in-vehicle networking interfaces such as CAN. Deployment on deeply embedded actuator ECUs would require additional optimization and validation, which we leave for future work.

VI. Evaluation

In this section, we evaluate the effectiveness of *SaferCAN* by the testbed introduced in Section. III. First, we present a detailed case study that illustrates the execution of *SaferCAN*, highlighting the intermediate outputs produced by each module and how they jointly lead to a tiered safety decision. Second, we conduct larger-scale experiments across diverse attack intensities and traffic conditions to quantitatively assess detection timeliness, safety coverage, and robustness.

A. Case Study

We begin with a car following driving scenario as usual to demonstrate how *SaferCAN* operates.

Scenario setup. In our experiments, the behavior of the lead vehicle is governed by the `Traffic Manager` (`TM`) module [39]. The Traffic Manager facilitates control over the vehicle's speed by allowing adjustments to a speed-limit parameter. As a result, the lead vehicle's motion is determined by a sequence of percentage values that specify the desired speed relative to a predefined reference.

The longitudinal acceleration of the lead vehicle is assumed to be bounded within a moderate range to reflect normal driving behavior. In particular, prior studies on automated driving comfort and longitudinal dynamics indicate that acceleration magnitudes around $\pm 2\,\mathrm{m/s^2}$ are commonly adopted as acceptable limits for typical car-following scenarios [40]. In this case, we set the resulting longitudinal acceleration of the lead vehicle within a bounded interval: $a_t \in [-2,\, 2]$ m/s^2.

Primary Model Determination with Disturbance. To determine an appropriate disturbance bound $\bar{w}(p)$ for the primary model, we analyze the empirical distribution of the output mismatch sequence $\delta(y_k)$ computed from a dataset containing approximately 5×10^4 samples. The sequence $\delta(y_k)$ represents the residual error between the measured output and the nominal model prediction. The upper panel of Figure 12 illustrates the time evolution of $\delta(y_k)$ together with its empirical $p = 98\%$ and $p = 99\%$ percentile envelopes. These percentile-based bounds provide a statistical characterization of the magnitude of typical and extreme deviations observed in the data. While the 99% percentile yields a more conservative disturbance bound, it is dominated by rare outliers and leads to an overly loose over-approximation of the system behavior. In this work, we select the 98% percentile as a practical trade-off between robustness and tightness. The corresponding percentile value is used to define the disturbance magnitude $\bar{\omega} = 3.8e^{-4}$, which serves as a uniform bound on the additive disturbance affecting the primary model.

To assess the fidelity of the proposed disturbance-aware model AlloM, we conduct a validation study using 100 s of nominal CAN traffic collected under regular driving conditions. A sliding window of length $N_w = 400$ samples (approximately 4 s) is shifted across the dataset. Within each window, the first 90% of samples (about 3.6 s) serve as the training segment used to update the model parameters, while the remaining 10% (about 0.4 s) constitute the prediction horizon over which the short-term evolution of the vehicle's longitudinal velocity is forecast. Figure 12 demonstrates that the disturbance envelope quantified by $\bar{w}$ is sufficiently expressive to account for deviations between the measured and predicted trajectories throughout the evaluation horizon.

Safety Index Construction from Reachable Sets. To evaluate the safety of the predicted system evolution within the delay horizon, we introduce a TTC-inspired safety index defined over the augmented state $\tilde{x} = [h,\ v_L,\ x_e]^\top$ as

$$s(\tilde{x}) = \ell^\top \tilde{x}, \qquad \ell = \begin{bmatrix} 1 & \tau_{safe} & -\tau_{safe} C_{e,d} \end{bmatrix}^\top . \tag{21}$$

which penalizes small inter-vehicle spacing relative to the short-horizon closing speed. In this formulation, unsafe states correspond to $s(\tilde{x}) < 0$. The geometric interpretation of this construction is illustrated in Figure 13. Figure 13(a) visualizes the reachable tube of the augmented system in the (h, v_L, x_e) space. where the shaded polyhedra indicate the subsets of $\mathcal{R}_k$ for which $s(\tilde{x}) < 0$. Figure 13(b) provides a detailed comparison between the reachable bounds of the TTC, the ground-truth TTC, and the TTC value computed from instantaneous measurements under DoS attacks. Due to the disruption of sensor updates, the observed TTC under DoS exhibits a pronounced stagnation behavior, remaining nearly constant over a time interval despite the continued closing of the inter-vehicle distance. This phenomenon reflects a typical failure mode of the conventional TTC computation, which relies on point-wise and potentially outdated measurements. In contrast, the TTC bounds derived from the reachable sets evolve continuously over time and consistently enclose the ground-truth TTC trajectory. By accounting for all admissible system evolutions under bounded uncertainty and delayed sensing, the reachable TTC bounds provide a conservative yet reliable characterization of the true collision risk, even when the observed TTC becomes misleading. Finally, Figure 13(c) demonstrates the resulting safety interpretation when the threshold is set to $\tau_{safe} = 0.75\,\mathrm{s}$, where the shaded polyhedra indicate the subsets of $\mathcal{R}_k$ for which $s(\tilde{x}) < 0$.

B. Evaluation under Different Lead-Vehicle Driving Scenarios and DoS Attack Intensities

To systematically evaluate the effectiveness of the proposed monitor under representative operating conditions, we consider a set of commonly encountered lead-vehicle driving scenarios together with different DoS attack intensities. Each experiment is decomposed into multiple disjoint temporal segments, where each segment corresponds to a continuous time interval affected by a DoS attack and is treated as an independent evaluation sample. For each segment, three types of time-to-collision information are considered:

- $\mathtt{TTC}_{\mathrm{gt}}$: the ground-truth TTC computed from the real-time simulator state;
- $\mathtt{TTC}_{\mathrm{dos}}$: the observed TTC derived from sensor measurements corrupted by DoS attacks;
- $\mathtt{TTC}_{\mathrm{rt}}$: the TTC reachable tube estimated by *SaferCAN*.

Throughout the evaluation, a fixed TTC threshold τ_{safe} is adopted to define safety-critical events uniformly across all types of TTC information.

Diverse Driving Scenario and Attack Intensities. To evaluate the proposed monitor under diverse operating conditions, we vary both the driving scenarios and the DoS attack intensities. Driving scenarios in our experiments correspond to distinct lead-vehicle velocity profiles, i.e., each scenario is defined by a different time series of the lead vehicle speed that induces specific longitudinal dynamics (e.g., near-steady cruising or aggressive speed variations). We generate these scenarios by editing the lead vehicle's velocity profile via the `TM` module and then replaying the modified profile in the co-simulation. Because `TM` prescribes velocity directly rather than acceleration, the resulting lead-vehicle acceleration is only approximately bounded; in practice, the acceleration is kept largely within $[-2, 2]\ \mathrm{m/s^2}$, although rare transient excursions beyond this range can occur. By varying the velocity profiles while maintaining these acceleration bounds, we obtain a set of scenarios that exhibit diverse longitudinal behaviors for robust evaluation of the proposed monitor. DoS attack intensity is controlled independently by adjusting the sleep parameter T_{sleep} of the attack injector.

The final evaluation comprises three distinct lead-vehicle scenarios and three DoS attack intensities. Each driving scenario is defined by a different lead-vehicle speed profile: the scenarios primarily differ in the frequency and amplitude of speed variations, thus producing diverse longitudinal dynamics for testing. For the attack axis, we consider three intensity levels (LOW, MEDIUM, and HIGH). As described in Section III-B, each intensity level corresponds to a range of the injector sleep parameter T_{sleep}, with smaller T_{sleep} values producing more aggressive flooding and stronger message contention on the virtual CAN bus. To account for run-to-run variation, every combination of scenario and attack intensity is executed independently three times. From these runs, we extract all DoS-affected segments and use them for evaluation and aggregation. In total, the experimental campaign produced $\mathbf{10{,}377}$ DoS-affected sample segments, which are summarized quantitatively in Table III.

Ground-Truth Labeling. Each segment is assigned a binary ground-truth label based on the $\mathtt{TTC}_{\mathrm{gt}}$. Specifically, if $\mathtt{TTC}_{\mathrm{gt}}$ within the segment violates the safety threshold τ_{safe}, the segment is labeled as *unsafe* $(+1)$; otherwise, it is labeled as *safe* (-1).

Alarm Output of *SaferCAN* Labeling. *SaferCAN* produces two levels of alarms, namely a *remind* (weak alarm) and a *warn* (strong alarm). Based on the safety index design introduced in Section V-E, the runtime output of *SaferCAN* is determined according to predefined thresholds on the maximum unsafe coverage ratio $r_{\max}$. A *remind* is issued when $r_{\max} > 0$, indicating that a non-zero portion of the reachable set intersects with the unsafe region. This condition captures early-stage risk, where unsafe outcomes are possible but do not yet dominate the future evolution. A *warn* is issued when $r_{\max} \geq 0.5$, indicating that unsafe states constitute a majority of the reachable set and therefore dominate the admissible future behaviors. For each segment, the presence or absence of an alarm determines the monitor's decision, which can be directly compared against the ground-truth label to categorize the outcome as a true positive, false positive, true negative, or false negative.

Timing Metrics. In addition to detection accuracy, we evaluate the temporal behavior of the proposed monitor, with a particular focus on its response time under DoS-induced sensing

TABLE III: Scenario-wise detection performance under different attack intensities. Metrics are reported separately for the *warn* (strong alarm) and *remind* (weak alarm) levels. $\overline{\Delta t}$ denotes the mean warning lead time relative to the ground-truth TTC.

Scenario	Attack Strength	Warn (strong alarm)				Remind (weak alarm)				Timing Metrics (seconds)		
		Prec.	Rec.	FPR	F1	Prec.	Rec.	FPR	F1	$\overline{\Delta t}_{\text{dos}}$	$\overline{\Delta t}_{\text{warn}}$	$\overline{\Delta t}_{\text{remind}}$
1	LOW	0.76	0.89	0.01	0.82	0.60	0.95	0.03	0.74	-0.1510	-0.0460	0.0976
	MEDIUM	0.86	0.85	0.01	0.85	0.64	0.96	0.05	0.77	-0.1689	-0.0197	0.0160
	HIGH	0.81	0.85	0.01	0.83	0.67	0.95	0.02	0.79	-0.2234	-0.0326	0.0752
2	LOW	0.62	0.86	0.01	0.73	0.52	0.94	0.02	0.67	-0.2111	-0.0134	0.0928
	MEDIUM	0.82	0.94	0.02	0.88	0.76	1.00	0.03	0.86	-0.1561	0.0149	0.1367
	HIGH	0.90	0.96	0.01	0.93	0.70	1.00	0.03	0.82	-0.3038	0.0265	0.1215
3	LOW	0.73	0.81	0.01	0.77	0.59	0.94	0.02	0.72	-0.0782	0.0568	0.0593
	MEDIUM	0.93	0.81	0.00	0.86	0.72	0.94	0.02	0.82	-0.1009	-0.0097	0.0557
	HIGH	0.74	0.80	0.01	0.77	0.42	0.97	0.05	0.59	-0.1633	-0.0075	0.0841

degradation. For each DoS-affected segment in which an alarm is triggered, we record the alarm times corresponding to both the *remind* and *warn* levels and derive three timing-related metrics. First, the DoS observation delay Δt_{dos} is defined as the time difference between the threshold-crossing time of the DoS-corrupted TTC observation and that of the ground-truth TTC. This metric characterizes the sensing delay introduced by the DoS attack and serves as a baseline reference for subsequent timing comparisons. Second, the warning lead time Δt_{warn} is defined as the time difference between the *warn* alarm raised by *SaferCAN* and the alarm time obtained from the ground-truth TTC using the same safety threshold. In this sense, the ground-truth TTC-based alarm acts as a baseline, and a positive value of Δt_{warn} indicates that *SaferCAN* issues a strong alarm earlier than the TTC-based reference. Third, the reminder lead time Δt_{remind} is defined analogously as the time difference between the *remind* alarm time and the corresponding ground-truth TTC-based alarm time. This metric reflects how early the weak alarm is triggered relative to the TTC-based baseline under ideal sensing conditions.

Interpretation of Detection Results. Table III summarizes the scenario-wise detection performance of *SaferCAN* under different attack intensities. Results are reported separately for the *warn* and *remind* levels, covering both detection accuracy and temporal characteristics. Overall, the *warn* level consistently achieves high precision and recall across all scenarios, with the false positive rate remaining at or below 0.01 in most cases. This indicates that strong alarms are both reliable and conservative, producing very few false positives even under high attack intensity. In contrast, the *remind* level exhibits higher recall but relatively lower precision, reflecting its intended role as a more sensitive early warning mechanism. Across different attack strengths, the detection performance remains largely stable. In particular, increasing attack intensity does not lead to a systematic degradation in the F1 score, especially for the *warn* level. This observation suggests that *SaferCAN* is robust to variations in DoS attack severity and is not overly sensitive to aggressive CAN flooding. Furthermore, the timing metrics $\overline{\Delta t}$ further demonstrate the effectiveness of the proposed approach. Meanwhile, Δt_{warn} and Δt_{remind} show that alarms are generally triggered close to, yet ahead of, the TTC-based reference. As expected, the *remind* level provides earlier warnings at the cost of a higher false positive rate, while the *warn* level offers more conservative but reliable alerts. Overall, these results demonstrate that *SaferCAN* achieves a favorable balance between early detection and reliability, enabling timely and robust responses to DoS attacks across diverse driving scenarios.

VII. Discussion

Assessment of *SaferCAN* under nonlinear Dynamics. In current design, *SaferCAN* uses a local data-driven linear model and a residual-based disturbance bound for reachable-tube computation. The residual bound is intended to over-approximate local modeling errors caused by nonlinear dynamics. Therefore, whether the reachable tube safely encloses the true future behavior depends on whether the recent trusted CAN data and validation residuals sufficiently represent the current driving regime.

A natural future improvement is to make the disturbance bound adaptive to the current operating regime. Specifically, *SaferCAN* can monitor online prediction residuals and enlarge or re-estimate the disturbance bound when the residual statistics deviate from those observed in the validation data. Another future direction is to replace the current lightweight reachable-tube computation module with a nonlinear reachability engine. In this extension, *SaferCAN* would still use the latest trusted CAN traffic to estimate the current state, control inputs, and uncertainty bounds, but the forward propagation would be performed on a richer nonlinear vehicle model instead of the current local linear approximation. For instance, nonlinear reachability engines such as Flow*'s Taylor-model flowpipe construction [41], CORA's set-based reachability for nonlinear systems [42], and sparse-polynomial-zonotope methods [43] could be used to propagate reachable sets for richer vehicle-dynamics models with bounded uncertainty. Such nonlinear reachability techniques could allow *SaferCAN* to better handle operating regimes with nonlinear effects. However, the replacement would also introduce additional runtime challenges, including higher computational cost, solver-parameter tuning, and accumulation of over-approximation error over the prediction horizon.

Extension of *SaferCAN* to CACC Systems. CACC differs from ACC in both controller structure and sensing/information

architecture. While ACC mainly relies on local sensing of the preceding vehicle, CACC additionally uses V2V information, such as predecessor or leader velocity, acceleration, or desired control input, to improve platoon performance and string stability [34], [44]. Therefore, the system model used for reachability analysis should be lifted from the single ego–lead car-following model to a platoon-level model that captures coupled inter-vehicle spacing dynamics, vehicle-level longitudinal dynamics, and the V2V predecessor/leader information used by the CACC controller.

The attack model would also need to be adapted. In CACC, the relevant communication-layer attack may occur on the inter-vehicle V2V channel. Such disruption changes the information-flow topology available to the cooperative controller and affects the cooperative feedforward path rather than only the local CAN sensing path [45]. Consequently, the attack-induced control-input model must be re-established for the target CACC controller. For example, under missing or delayed V2V information, a CACC controller may hold the last received cooperative input, fall back to ACC mode, switch to another information-flow topology, use prediction, or reconstruct the cooperative signal through redundant communication and data fusion. These controller-specific responses determine the degraded longitudinal command and therefore must be identified experimentally or derived analytically before reachable-tube propagation.

With these adaptations, *SaferCAN* could propagate finite-horizon reachable tubes to over-approximate possible platoon evolutions under communication-layer attacks. The resulting reachable platoon states can then be checked against CACC-specific safety indices, including minimum inter-vehicle distance, bounded spacing error, disturbance amplification, and string stability [46]. Compared with the current ACC setting, this extension would introduce additional modeling and computational challenges because of the higher-dimensional platoon state, V2V information-flow topology, and potentially nonlinear cooperative-control dynamics. We leave a full CACC instantiation and evaluation of *SaferCAN* for future work.

VIII. Related Work

CAN Bus IDS against DoS Attack. A large body of CAN IDS has been developed with a primary focus on detecting DoS attacks. Representative approaches analyze traffic-level characteristics such as message frequency, inter-arrival time, and entropy to identify abnormal bus load conditions [47], [48]. More recent work improves the robustness of DoS detection by exploiting refined timing models and statistical measures. Time-interval and conditional-entropy-based IDSs are proposed to better distinguish malicious flooding from benign bursty traffic [10]. In addition, CANival combines a time-interval-based analyzer with a signal-based analyzer to improve detection coverage across diverse CAN attacks [11].

Control-Theoretic Security and Safety Analysis. Control-oriented studies incorporate cyberattack effects into system models to analyze state estimation, closed-loop stability, and physical safety. For ACC systems, Qiu et al. model DoS as an interruption of sensing and control communication and analyze closed-loop stability and resilient sampling strategies [12]. Berdich and Groza directly impose replay, fuzzing, and DoS effects on Simulink models of ACC and AEB systems and evaluate their collision and injury consequences [13]. Beyond these ACC-specific studies, broader control-theoretic research has investigated secure state estimation and stability under cyberattacks in networked control systems. Wang et al. propose a distributed adaptive saturation method to reconstruct system states from measurements corrupted by sparse sensor attacks [49]. Doostmohammadian and Meskin study finite-time stability under intermittent DoS-induced communication outages, deriving sufficient conditions on the attack frequency and duration under an event-triggered control scheme [50].

State-Aware Detection and Defense in Automotive Systems. A growing body of work incorporates vehicle or driving state information into intrusion detection and defense mechanisms. Early context-aware approaches correlate CAN messages with vehicle operating conditions or control constraints to detect implausible behaviors [51], [52]. Control-invariant-based approaches monitor consistency between control commands and vehicle dynamics to identify malicious interference, particularly in robotic or automated vehicles [53], [14]. More recently, state-aware defense mechanisms have been proposed to actively prevent unsafe behaviors. SAID introduces a state-aware defense framework that integrates vehicle dynamics models with rule-based constraints to defend against message injection attacks on the CAN bus [15].

IX. Conclusion

In this paper, we constructed a co-simulation testbed with realistic CAN arbitration, enabling controllable DoS attacks to be launched during ACC operation. Under nominal steady-state operation, we analyze the delayed-sensing effect on local regulation stability and show that the delay induced by realistic CAN DoS attacks remains within the admissible stability margin of the OpenPilot-based ACC loop. Under transient car-following operation, we propose *SaferCAN*, a passive state-aware runtime safety assessment framework that models attack-induced control-input evolution and propagates finite-horizon reachable state tubes under bounded uncertainty to quantify future safety-margin degradation. Experimental results across different lead-vehicle motion profiles and CAN DoS intensities show that *SaferCAN* provides timely safety warnings with low false-positive rates and high recall. Overall, our results demonstrate that CAN DoS attacks may preserve local ACC stability while still inducing safety-critical risks under dynamic driving conditions.

References

[1] A. Moawad, M. Zebiak, J. Han, D. Karbowski, Y. Zhang, and A. Rousseau, "Effect of adaptive cruise control on fuel consumption in real-world driving conditions," *Nature Communications*, vol. 15, no. 1, p. 10016, 2024.

[2] PR Newswire, "Adaptive cruise control (acc) system market to reach $12.7 billion by 2034," https://www.prnewswire.com/news-releases/adaptive-cruise-control-acc-system-market-to-reach-12-7-billion-globally-by-2034-at-8-5-cagr-allied-market-research-302612971.html, 2024, allied Market Research report, accessed 2025.

[3] F. Bu and C.-Y. Chan, "Adaptive and cooperative cruise control," in *Handbook of Intelligent Vehicles*. Springer, 2012, pp. 191–207.

[4] A. Buscemi, I. Turcanu, G. Castignani, R. Crunelle, and T. Engel, "Canmatch: a fully automated tool for can bus reverse engineering based on frame matching," *IEEE Transactions on Vehicular Technology*, vol. 70, no. 12, pp. 12 358–12 373, 2021.

[5] C. Miller and C. Valasek, "Remote exploitation of an unaltered passenger vehicle," *Black Hat USA*, vol. 2015, no. S 91, pp. 1–91, 2015.

[6] P. Jing, Z. Cai, Y. Cao, L. Yu, Y. Du, W. Zhang, C. Qian, X. Luo, S. Nie, and S. Wu, "Revisiting automotive attack surfaces: A practitioners' perspective," in *2024 IEEE Symposium on Security and Privacy (SP)*. IEEE, 2024, pp. 2348–2365.

[7] M. D. Pesé, B. Gozubuyuk, E. Andrechek, H. Olufowobi, M. Hamad, and K. G. Shin, "Michican: Spoofing and denial-of-service protection using integrated can controllers," in *2025 55th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)*. IEEE, 2025, pp. 443–456.

[8] B. Lampe and W. Meng, "can-train-and-test: A new can intrusion detection dataset," in *2023 IEEE 98th Vehicular Technology Conference (VTC2023-Fall)*. IEEE, 2023, pp. 1–7.

[9] S. Rajapaksha, H. Kalutarage, G. Madzudzo, A. Petrovski, and M. O. Al-Kadri, "CAN-MIRGU: A comprehensive CAN bus attack dataset from moving vehicles for intrusion detection system evaluation," in *Proceedings of the Symposium on Vehicle Security and Privacy (VehicleSec)*, 2024, pp. 1–11.

[10] Z. Yu, Y. Liu, G. Xie, R. Li, S. Liu, and L. T. Yang, "Tce-ids: Time interval conditional entropy-based intrusion detection system for automotive controller area networks," *IEEE Transactions on Industrial Informatics*, vol. 19, no. 2, pp. 1185–1195, 2022.

[11] H. Kang, T. Vo, H. K. Kim, and J. B. Hong, "Canival: A multimodal approach to intrusion detection on the vehicle can bus," *Vehicular Communications*, vol. 50, p. 100845, 2024.

[12] Q. Tianxiang, H. Defeng, L. Liangye, and S. Xiulan, "Adaptive cruise control of vehicles subject to denial-of-service," in *2017 32nd Youth Academic Annual Conference of Chinese Association of Automation (YAC)*. IEEE, 2017, pp. 382–386.

[13] A. Berdich and B. Groza, "Cyberattacks on adaptive cruise controls and emergency braking systems: Adversary models, impact assessment, and countermeasures," *IEEE Transactions on Reliability*, vol. 73, no. 2, pp. 1216–1230, 2024.

[14] R. Quinonez, J. Giraldo, L. Salazar, E. Bauman, A. Cardenas, and Z. Lin, "{SAVIOR}: Securing autonomous vehicles with robust physical invariants," in *29th USENIX security symposium (USENIX Security 20)*, 2020, pp. 895–912.

[15] L. Xue, Y. Liu, T. Li, K. Zhao, J. Li, L. Yu, X. Luo, Y. Zhou, and G. Gu, "{SAID}: State-aware defense against injection attacks on in-vehicle network," in *31st USENIX Security Symposium (USENIX Security 22)*, 2022, pp. 1921–1938.

[16] Comma.ai, "Openpilot," https://github.com/commaai/openpilot, 2025.

[17] A. Dosovitskiy, G. Ros, F. Codevilla, A. Lopez, and V. Koltun, "Carla: An open urban driving simulator," in *Conference on Robot Learning*. PMLR, 2017, pp. 1–16.

[18] H. J. Jo and W. Choi, "A survey of attacks on controller area networks and corresponding countermeasures," *IEEE Transactions on Intelligent Transportation Systems*, vol. 23, no. 7, pp. 6123–6141, 2021.

[19] H. S. Foroush and S. Martinez, "On event-triggered control of linear systems under periodic denial-of-service jamming attacks," in *2012 IEEE 51st IEEE Conference on Decision and Control (CDC)*. IEEE, 2012, pp. 2551–2556.

[20] R. Song, M. O. Ozmen, H. Kim, R. Muller, Z. B. Celik, and A. Bianchi, "Discovering adversarial driving maneuvers against autonomous vehicles," in *32nd USENIX Security Symposium (USENIX Security 23)*, 2023, pp. 2957–2974.

[21] SAE International, "SAE J3016: Taxonomy and definitions for terms related to driving automation systems for on-road motor vehicles," https://www.sae.org/standards/j3016_202104-taxonomy-definitions-terms-related-driving-automation-systems-road-motor-vehicles, 2021, accessed: 2026-05-04.

[22] K.-T. Cho and K. G. Shin, "Error handling of in-vehicle networks makes them vulnerable," in *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, 2016, pp. 1044–1055.

[23] AUTOSAR, "Description of the autosar standard errors," AUTOSAR, Tech. Rep. AUTOSAR CP R24-11, 2024.

[24] NXP Semiconductors, "Automotive gateway for the secure connected car," NXP Semiconductors, Tech. Rep., 2017.

[25] X. Zhou, A. Schmedding, H. Ren, L. Yang, P. Schowitz, E. Smirni, and H. Alemzadeh, "Strategic safety-critical attacks against an advanced driver assistance system," in *2022 52nd Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)*. IEEE, 2022, pp. 79–87.

[26] Texas Instruments, *TMS320C28x CAN Module Reference Guide*, 2015, literature Number: SPRU074F. [Online]. Available: https://www.ti.com/lit/ug/spru074f/spru074f.pdf

[27] H. Song and H. Kim, "Car-hacking datasest for the intrusion detection," *Accessed: Sep*, vol. 6, 2020.

[28] G. Orosz, "Connected cruise control: modelling, delay effects, and nonlinear behaviour," *Vehicle System Dynamics*, vol. 54, no. 8, pp. 1147–1176, 2016.

[29] V. B. Kolmanovskii and V. R. Nosov, *Stability of functional differential equations*. Elsevier, 1986, vol. 180.

[30] A. Thowsen, "The routh-hurwitz method for stability determination of linear differential-difference systems," *International Journal of Control*, vol. 33, no. 5, pp. 991–995, 1981.

[31] S. James and S. R. Anderson, "Linear system identification of longitudinal vehicle dynamics versus nonlinear physical modelling," in *2018 UKACC 12th International Conference on Control (CONTROL)*. IEEE, 2018, pp. 146–151.

[32] Y. Ruan, C. Zhao, Z. Yang, Y. Shu, P. Cheng, and J. Chen, "Picacan: Reverse engineering physical semantics of signals in can messages using physically-induced causalities," *IEEE Transactions on Mobile Computing*, 2025.

[33] C.-Y. Chen, K. G. Shin, and S. Dadras, "Context-aware anomaly detection using vehicle dynamics," in *Proceedings of the 27th International Symposium on Research in Attacks, Intrusions and Defenses*, 2024, pp. 531–545.

[34] Z. Wang, G. Wu, and M. J. Barth, "A review on cooperative adaptive cruise control (cacc) systems: Architectures, controls, and applications," in *2018 21st International Conference on Intelligent Transportation Systems (ITSC)*. IEEE, 2018, pp. 2884–2891.

[35] R. Nabati and H. Qi, "Radar-camera sensor fusion for joint object detection and distance estimation in autonomous vehicles," *IEEE Transactions on Intelligent Transportation Systems*, vol. 22, no. 6, pp. 3650–3661, 2021.

[36] B. Ciuffo, V. Punzo, and M. Montanino, "Thirty years of gipps' car-following model: Applications, developments, and new features," *Transportation Research Record*, vol. 2315, no. 1, pp. 89–99, 2012.

[37] M. M. Minderhoud and P. H. Bovy, "Extended time-to-collision measures for road traffic safety assessment," *Accident Analysis & Prevention*, vol. 33, no. 1, pp. 89–97, 2001.

[38] S. Liu, X. Wang, O. Hassanin, X. Xu, M. Yang, D. Hurwitz, and X. Wu, "Calibration and evaluation of responsibility-sensitive safety (rss) in automated vehicle performance during cut-in scenarios," *Transportation Research Part C: Emerging Technologies*, vol. 125, p. 103037, 2021.

[39] S. Malik, M. A. Khan, and H. El-Sayed, "Carla: Car learning to act—an inside out," *Procedia Computer Science*, vol. 198, pp. 742–749, 2022.

[40] J. Ossig, S. Cramer, and K. Bengler, "Longitudinal acceleration during lane changes-a human-centered investigation for automated driving," in *14. Uni-DAS eV Workshop Fahrerassistenz and automatisiertes Fahren*, 2022.

[41] X. Chen, E. Ábrahám, and S. Sankaranarayanan, "Flow*: An analyzer for non-linear hybrid systems," in *International Conference on Computer Aided Verification*. Springer, 2013, pp. 258–263.

[42] M. Althoff, "An introduction to cora 2015," in *Proceedings of the Workshop on Applied Verification for Continuous and Hybrid Systems*, 2015, pp. 120–151.

[43] N. Kochdumper and M. Althoff, "Sparse polynomial zonotopes: A novel set representation for reachability analysis," *IEEE Transactions on Automatic Control*, vol. 66, no. 9, pp. 4043–4058, 2021.

[44] J. Ploeg, B. T. Scheepers, E. Van Nunen, N. Van de Wouw, and H. Nijmeijer, "Design and experimental evaluation of cooperative adaptive cruise control," in *2011 14th International IEEE Conference on Intelligent Transportation Systems (ITSC)*. IEEE, 2011, pp. 260–265.

[45] S. Gong, A. Zhou, J. Wang, T. Li, and S. Peeta, "Cooperative adaptive cruise control for a platoon of connected and autonomous vehicles considering dynamic information flow topology," in *2018 21st International Conference on Intelligent Transportation Systems (ITSC)*, 2018, pp. 185–190.

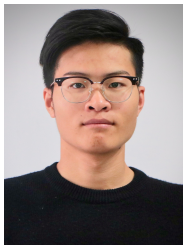
[46] J. A. Ligthart, J. Ploeg, E. Semsar-Kazerooni, M. Fusco, and H. Nijmeijer, "Safety analysis of a vehicle equipped with cooperative adaptive cruise control," *IFAC-PapersOnLine*, vol. 51, no. 9, pp. 367–372, 2018.

[47] M. Müter and N. Asaj, "Entropy-based anomaly detection for in-vehicle networks," in *2011 IEEE Intelligent Vehicles Symposium (IV)*. IEEE, 2011, pp. 1110–1115.

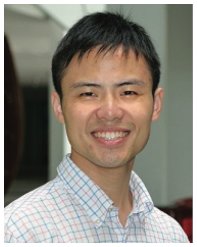
[48] Y. Liu, L. Xue, S. Wang, X. Luo, K. Zhao, P. Jing, X. Ma, Y. Tang, and H. Zhou, "Vehicular intrusion detection system for controller area network: A comprehensive survey and evaluation," *IEEE Transactions on Intelligent Transportation Systems*, 2025.
[49] Q. Wang, J. Zhang, Y. Qian, and A.-Y. Lu, "Secure state estimation against sparse sensor attacks: A distributed adaptive saturation method," *IEEE Transactions on Automation Science and Engineering*, vol. 23, pp. 2091–2099, 2025.
[50] M. Doostmohammadian and N. Meskin, "Finite-time stability under denial of service," *IEEE Systems Journal*, vol. 15, no. 1, pp. 1048–1055, 2021.
[51] A. Wasicek, M. D. Pesé, A. Weimerskirch, Y. Burakova, and K. Singh, "Context-aware intrusion detection in automotive control systems," in *Proc. 5th ESCAR USA Conf*, 2017, pp. 21–22.
[52] M. Müter, A. Groll, and F. C. Freiling, "A structured approach to anomaly detection for in-vehicle networks," in *2010 Sixth International Conference on Information Assurance and Security*. IEEE, 2010, pp. 92–98.
[53] H. Choi, W.-C. Lee, Y. Aafer, F. Fei, Z. Tu, X. Zhang, D. Xu, and X. Deng, "Detecting attacks against robotic vehicles: A control invariant approach," in *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*, 2018, pp. 801–816.

Yucheng Ruan received the B.E. degree in Mehcanical Engineering, from Zhejiang University, Hangzhou, China. He is currently working toward the PhD degree in Control Science and Engineering of Zhejiang University. His research interests include cyber-physical system security.

Chengcheng Zhao (Member, IEEE) received the Ph.D. degree in control science and engineering in 2018 from Zhejiang University, Hangzhou, China. She worked as a postdoctoral fellow at the ECE department, the University of Victoria, from 2019 to 2020. Currently, she is a ZJU100 Young Professor in the College of Control Science and Engineering at Zhejiang University. Her research interests include Security and Privacy in Cyber-Physical Systems, Security and Safety in Autonomous Systems, etc. She received the IEEE PESGM 2017 Best Conference Paper award, and one of her papers was selected as a finalist for the IEEE ICCA 2017 Best Student Paper Award. She serves as an Associate Editor for the IEEE Transactions on Vehicular Technology.



Zeyu Yang received the Ph.D. degree in control science and engineering in 2022 from Zhejiang University, Hangzhou, China. He is currently a Research Fellow with iTrust, Singapore University of Technology and Design, Singapore. His research interests include cyber-physical system security and industrial control system security.



Peng Cheng (Member, IEEE) is currently executive dean, and Changjiang Chair Professor for college of control science and engineering, Zhejiang University. He serves/served as Associate Editors for the IEEE Transactions on Control of Network Systems, IEEE Transactions on Cloud Computing. He also served as Guest Editors for IEEE Transactions on Automatic Control and IEEE Transactions on Signal and Information Processing over Networks. His research interests include control system security, cyber-physical systems, and cloud networking.

Jiming Chen (Fellow, IEEE) received B.Sc degree and Ph.D degree both in Control Science and Engineering from Zhejiang University. He was a visiting researcher at University of Waterloo from 2008 to 2010. Currently he is Professor with College of Control Science and Engineering, Zhejiang University. His research interests include IoT, networked control, and wireless networks.

Kasper Rasmussen (Senior Member, IEEE) got his Ph.D. at ETH Zurich where he worked mainly on security issues relating to secure time synchronization and secure localization with a particular focus on distance bounding. After completing his Ph.D., Kasper worked as a post-doc at University of California, Irvine, before joining the University of Oxford in 2013. In 2015 he was awarded a University Research Fellowship from the Royal Society in London. Kasper is now Professor of Information Security in the Computer Science Department at the University of Oxford, where he leads a research group that works on different aspects of system and communication security, including Security of Wireless Networks, Protocol design, Applied Cryptography, Security of embedded systems and Cyber-physical systems.