Attack Impact and Misbehavior Detection in Vehicular Platoons

Konstantinos Kalogiannis
KTH Royal Institute of Technology
Stockholm, Sweden
konkal@kth.se

Mohammad Khodaei
KTH Royal Institute of Technology
Stockholm, Sweden
khodaei@kth.se

Weaam Mostafa Nemr Mohamed Bayaa
KTH Royal Institute of Technology
Stockholm, Sweden
bayaa@kth.se

Panos Papadimitratos
KTH Royal Institute of Technology
Stockholm, Sweden
papadim@kth.se

ABSTRACT

Cooperative Adaptive Cruise Control (CACC), a promising Vehicular Ad-hoc Network (VANET) application, automates transportation and improves efficiency. Vehicles form a platoon, following a leader, with their controllers automatically adjusting velocity, based on messages by other vehicles, to keep appropriate distances for safety. Towards deploying secure CACC, several proposals in academia and standardization leave significant questions unanswered. Thwarting adversaries is hard: cryptographic protection ensures access control (authentication and authorization) but falsified kinematic information by faulty insiders (platoon members with credentials, even the platoon leader) can cause platoon instability or vehicle crashes. Filtering out such adversarial data is challenging (computational cost and high false positive rates) but, most important, state-of-the-art misbehavior detection algorithms completely fail during platoon maneuvering. In this paper, we systematically investigate how and to what extent controllers for existing platooning applications are vulnerable, mounting a gamut of attacks, ranging from falsification attacks to jamming and collusion; including two novel attacks during maneuvering. We show how the existing middle-join and leave processes are vulnerable to falsification or 'privilege escalation' attacks. We mitigate such vulnerabilities and enable vehicles joining and exiting from any position (middle-join and middle-exit). We propose a misbehavior detection system that achieves an F_1 score of ≈87% on identifying attacks throughout the lifetime of the platoon formation, including maneuvers. Our cyberphysical simulation framework can be extended to assess any other driving automation functionality in the presence of attackers.

CCS CONCEPTS

• **Networks** → **Network security**; • **Security and privacy** → **Distributed systems security**; • **Computing methodologies** → **Simulation tools**; **Anomaly detection**.



This work is licensed under a Creative Commons Attribution-NonCommercial-ShareAlike International 4.0 License.

WiSec '22, May 16–19, 2022, San Antonio, TX, USA
© 2022 Copyright held by the owner/author(s).
ACM ISBN 978-1-4503-9216-7/22/05.
https://doi.org/10.1145/3507657.3528552

KEYWORDS

Falsification Attacks, Connected Vehicles, Internal Adversaries, Platoon Maneuvers, Misbehavior Detection, Hidden Markov Models

ACM Reference Format:
Konstantinos Kalogiannis, Mohammad Khodaei, Weaam Mostafa Nemr Mohamed Bayaa, and Panos Papadimitratos. 2022. Attack Impact and Misbehavior Detection in Vehicular Platoons. In *Proceedings of the 15th ACM Conference on Security and Privacy in Wireless and Mobile Networks (WiSec '22), May 16–19, 2022, San Antonio, TX, USA.* ACM, New York, NY, USA, 15 pages. https://doi.org/10.1145/3507657.3528552

1 INTRODUCTION

Vehicle-to-Vehicle (V2V) and Vehicle-to-Infrastructure (V2I) communication seek to enhance transportation safety and efficiency: vehicles disseminate Cooperative Awareness Messages (CAMs) and Decentralized Environmental Notification Messages (DENMs). As is well-understood that Vehicular Communication (VC) systems are vulnerable to attacks, security solutions have been developed by standardization bodies (IEEE 1609.2 WG [3] and ETSI [26]) and projects (SeVeCom [51, 53], and CAMP [73]). To secure V2V and V2I (V2X) communication, Public Key Cryptography (PKC) is used: a set of Certification Authorities (CAs) constitutes the Vehicular Public-Key Infrastructure (VPKI), e.g., [41, 73], providing multiple anonymized certificates, termed *pseudonyms*, to each legitimate vehicle. Vehicles switch pseudonyms (not previously used) towards unlinkable digitally signed CAMs/DENMs.

Vehicular platooning promises reduced fuel consumption [13, 23, 42], more efficient transportation (i.e., improved throughput with the existing infrastructure) [7, 17, 46, 72, 77, 78], and safer and more comfortable driving [71]. V2X communication leads to platoon topologies as those illustrated in Fig. 1, enabling Cooperative Adaptive Cruise Control (CACC) [37]. A variety of controllers aim at providing *local stability* and *string stability* [21, 72]: maintaining distances between each vehicle and its successor/predecessor with appropriate margins, and determining how spacing errors propagate through the platoon. To achieve this, controllers leverage CAMs/DENMs and radar measurements of adjacent vehicles [48]. Several on-going projects and Field Operational Tests (FOTs), e.g., [7, 43, 46], as well as pre-standardization work [27], investigate the case of platooning. In a platooning environment (level 3 or above as per the Society of Automotive Engineers (SAE)-J3016 standard [2]), vehicles interact with the platoon leader and request to join, typically, from the platoon tail [17].

Modern platooning schemes allow vehicles joining from any position, namely performing *middle-join* and *middle-exit* maneuvers. This has several advantages: sorting vehicles out based on their trajectory paths and other attributes, i.e., engine and braking capabilities, results in fewer split and merge maneuvers. This results in increased platoon stability and safety, and higher road utilization (throughput) [25, 33, 45].

While the benefits of platooning are evident, deploying such a high-stake application, even at the lowest level of automation, cannot materialize unless all its underlying processes and operations are secure and reliable [40]. Mitigating misbehavior by internal adversaries is hard: cryptographic protocols ensure access control (authentication and authorization) but internal adversaries launching falsification attacks [12, 69] can destabilize the platoon and jeopardize passenger/driver safety. Worse even, a compromised platoon leader disseminating false *position*, *speed* and *acceleration* values to all members of the platoon, e.g., [35], could cause a collision; exactly because controllers mostly rely on leader-provided data, the attack impact would be higher [35]. To alleviate the effects of these attacks, vehicles can operate with larger intra-platoon distances, thus, increasing the reaction times (of the platoon members) in the presence of a misbehavior. However, operating platoons and CACC with larger vehicle distances cannot eliminate the collision risk. This would not only make platooning less efficient, but also fail mitigating attacks and eradicating vehicle collisions [35, 69].

Attacks can be more sophisticated: a compromised insider could initiate a data falsification attack and collude with an external entity, e.g., a drone flying above the platoon performing a clogging Denial of Service (DoS) [18] or a jamming attack [69]. This type of collusion, not yet shown in the literature, is difficult to mitigate as the external entity cannot be held accountable.

The trend towards CACC maneuvering and improved platooning is a double-edged sword: it gives an internal adversary new ways to destabilize the platoon formation (in particular, the joining vehicle and its followers), based on appropriately crafted falsified V2X messages. Furthermore, the adversary could exploit the execution of middle-join and middle-exit protocols [17, 29] towards a *privilege escalation* attack. When the platoon is split to perform the maneuvers, a platoon member (positioned after the joiner/leaver) would inadvertently be promoted to lead the second formation (until the platoons re-merge). But, if a compromised vehicle is elevated to be a leader, it can more easily destabilize the platoon, even catastrophically. These scenarios need to be properly analyzed, notably in terms of their physical impact. More so, appropriate secure CACC maneuvering protocols are required.

Plausibility checks of the position, speed and acceleration data received in V2V messages [38] can filter out a range of falsified data, but they cannot detect carefully crafted attacks, e.g., a falsified yet gradual increase in speed within plausible ranges. Other Misbehavior Detection Schemes (MDSs), leveraging data fusion and/or Kalman Filters [16, 31, 35], aim at thwarting data falsification attacks. However, they are limited in assuming that at all times specific neighbors and/or platoon members are benign, e.g., other vehicles on the road [16] or the platoon leader [31]. More so, existing MDSs cannot discern maneuvering processes from misbehavior [35], as their model relies on static 'normal' behavior; thus, any benign deviation from it, notably a maneuver, would be flagged as misbehavior. This would disrupt the functionality of platooning and diminish its benefits. Therefore, we need to design MDSs that reliably detect sophisticated attacks, without mis-classifying maneuvering as misbehavior.

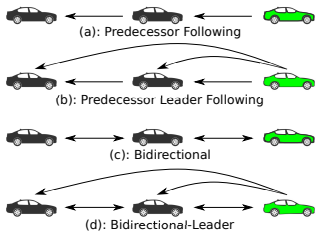


Figure 1: Information Flow Topologies.

Contributions: We introduce novel vehicular platooning attacks: falsification attacks during the join and leave maneuver processes, and falsification attacks combined with targeted jamming (with colluding adversaries). We design secure middle-join and leave protocols to mitigate the privilege escalation vulnerability. We evaluate the effectiveness of a state-of-the-art MDS at mitigating attacks in a dynamic platooning environment and we propose a new MDS, leveraging machine learning algorithms. Unlike existing MDSs, our scheme discerns misbehavior from maneuvering. We significantly extend existing simulation tools, integrating the above elements into *Artemis*, a comprehensive framework to analyze maneuvers, attack scenarios, misbehavior detection schemes and mitigation techniques.

In the rest of the paper, we describe the background (Sec. 2), related works (Sec. 3), adversarial model and the attack procedures (Sec. 4). We follow up with our proposals to detect attacks (Sec. 5), the experimental evaluation setup (Sec. 6), and the attack (Sec. 7) and countermeasure (Sec. 8) analyses, before we conclude.

2 BACKGROUND

System Overview: Vehicles forming a platoon, reach a consensus to elect a leader, e.g., based on a voting mechanism [36] or based on the history of being a leader [19]. The *platoon leader* facilitates vehicles joining the platoon by periodically disseminating the platoon identity, status, and destination [27]. Vehicles interact with the leader and request to join; typically, from the platoon tail [17]. The leader is responsible for accepting or rejecting join requests, depending on the situation; e.g., rejecting a join request if the road traffic condition does not allow extra spacing. Once accepted, the *joiner* vehicle positions itself behind the platoon tail and notifies the leader; which in turn informs all platoon members about the new joiner. Platoon joins can be performed by one car at a time, or by merging two platoons, e.g., for road safety reasons [10].

Information flows within a platoon [76] as illustrated in Fig. 1. Each platoon member communicates in four different ways: (i) unidirectionally, receiving from its predecessor (Fig. 1.a), (ii) unidirectionally, receiving from its predecessor and the leader (Fig. 1.b), (iii) bidirectionally, with the predecessor (Fig. 1.c), and (iv) bidirectionally, with the predecessor and unidirectionally receiving from the leader (Fig. 1.d). Predecessor-Leader topologies are typically stable because the effective time for a V2V message (i.e., CAM, DENM)

Table 1: Comparing Controllers Required Information: Distance (𝔻), Speed (𝕊), Acceleration (𝔸) and Position (ℙ); Radar Measurements (ℝ) and V2V Communication (𝕍).

Controller	Policy	Predecessor			Leader			Topology
		𝔻	𝕊	𝔸	ℙ	𝕊	𝔸	
ACC	CTH	ℝ	ℝ	–	–	–	–	–
PATH	CVS	ℝ	𝕍ℝ	𝕍	–	𝕍	𝕍	PL Following
Consensus (CNSS)	BOTH	ℝ	–	–	𝕍	𝕍	–	PL Following
Flatbed (FLBD)	CVS	ℝ	𝕍ℝ	–	–	𝕍	–	Leader Following
Ploeg (PLG)	CTH	ℝ	ℝ	𝕍	–	–	–	Predecessor Following

to reach any vehicle is short, with the leader reaching every platoon member directly (rather than relying on a multihop transmission), thus improving the string stability.

There are two main spacing *policies*: Constant Vehicle Spacing (CVS) and Constant Time Headway (CTH) [27, 28]. CVS is enforced by utilizing speed and acceleration information obtained from one's predecessor. CVS tends to create smaller space among platoon members, thus achieving higher traffic throughput [67] by trading off platoon stability under certain circumstances, e.g., sudden braking. CTH uses the current speed and the *headway*, the time it would take for the front of a vehicle to reach the rear of its predecessor, to calculate the intra-platoon distance. Higher speeds for a given headway value imply larger intra-platoon distances, leading to reduced traffic throughput (due to larger vehicle distances) [32].

Table 1 compares different controllers in terms of their policies and topologies. Adaptive Cruise Control (ACC) is included for completeness, even though it is not used for platooning. The *PATH* controller [58] relies on *Predecessor-Leader* topology and CVS policy (Fig. 1.b), with vehicles receiving the speed and acceleration from their leader and predecessor, or obtain them based on own radar measurements. *Consensus* [61] also relies on the Predecessor-Leader topology but uses information only from the leader, its position and speed, and enforces a combination of CVS and CTH. *Flatbed* [14] uses the speeds of the leader and the predecessor (with smaller weight), obtained based on a Predecessor-Leader topology, to maintain its CVS policy. Flatbed simulates a, tow truck, that pulls the whole platoon: the speed difference between the truck and each vehicle is used to calculate the correct acceleration and intra-platoon distances. Finally, *Ploeg* [56] uses a *Predecessor Following* topology and utilizes the predecessor acceleration to achieve its CTH policy.

Hidden Markov Models Fundamentals: Markov models enable predicting the future state of a system based on its current state. A Hidden Markov Model (HMM) is based on two stochastic processes: an observable process that represents the sequence of observations, and a hidden process that can be inferred indirectly by analyzing the observation sequence, e.g., transitions from one observed state to another. Hidden states transitions are governed by a set of probabilities determined during the training phase.

An HMM consists of a set of N possible states, $S = \{S_1, S_2, \ldots, S_N\}$, and a matrix, π, containing the initial state probabilities, i.e., the probability of being at a specific initial hidden state. Given a hidden sequence $\{z = z_1, z_2, \ldots, z_T\}$, the initial matrix π is:

$$\pi_i = P(z_1 = S_i), \quad i = 1, \ldots, N \tag{1}$$

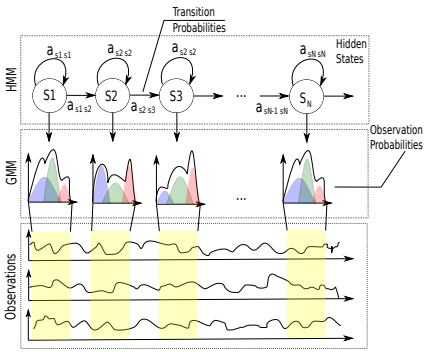


Figure 2: A Visualization of an HMM and GMM with Continuous Observations.

In addition, matrix $a = \{a_{ij}\}$ consists of the probabilities of transitioning from state S_i to S_j, where i, j = 1,. . . , N:

$$a_{i,j} = P(z_{t+1} = S_i | z_t = S_j) \tag{2}$$

The so-called *emission probability distribution*, B = {bi(·)}, represents the observation probability distribution at *state i*. For discrete probabilities, the observations belong to a codebook $V = \{v_1, v_2, \ldots, v_K\}$; given an observed time series $\{x = x_1, x_2, \ldots, x_T\}$, the b matrix is defined as:

$$b_i(x_t) = P(x_t = v_k | z_t = S_i) \tag{3}$$

If continuous, e.g., a Gaussian distribution or a mixture of multiple Gaussians, matrix b is defined as:

$$b_i(x_t) = \sum_{m=1}^{M} C_{i,m} \mathcal{S}(x_t) | \mu_{i,m}, \sum i, m \tag{4}$$

where 1≤i≤S, M is the number of Gaussians, μ is the mean, $\sum$ the covariance and $C_{i,m}$ the mixture coefficient where $\sum_{m=1}^{M} C_{i,m} = 1$.

We use a probability density function, termed Gaussian Mixture Model (GMM), a mixture of multiple multivariate Gaussian densities. GMM models the dependent relationship among multiple variables, e.g., speed and acceleration. Fig. 2 illustrates the HMM, the GMM, and the observations. Inference of the sequence of the hidden states is achieved by observing visible system changes. For example, in a platoon formation, increasing the intra-platoon gap could imply decreasing the speed (i.e., the observation) of a vehicle during a join maneuver (i.e., a sequence of hidden states).

3 RELATED WORK

It is intuitive that a malicious platoon leader can cause more harm than a malicious platoon member: a leader falsification attack would result in collision [35], while, with the same setup, a platoon member attack would either have no effect or it would destabilize the platoon. The effects of jamming on the string stability of the platoon were evaluated [15, 24, 69]. A drone, hovering above, initiates a jamming attack when the platoon is accelerating (without maneuvering). The drone effectively interrupts all the V2V communications towards destabilizing the formation; this results in a collision with the CVS policy, which utilizes the information from the predecessor to ensure constant gaps among members. The larger the

intra-platooning gaps, the lower the probability of collision, but the lower the platoon stability too. In our investigation, we propose novel collusion attacks and we go far beyond existing limited attack evaluations: we compare the severity of falsification attacks, including new variants, considering all state-of-the-art controllers.

Platoon management protocols, capable of performing several maneuvers, including join, leave, merge and split [9, 17, 18], initially allowed only joining from the tail of a platoon. However, there are advantages in enabling a middle join process: vehicles with the same, or nearby, destinations can be placed together, resulting in less maneuvers throughout the lifetime of the platoon [25, 33], thus, higher platoon stability (and efficiency). More so, vehicles with different engine and breaking capabilities can be positioned to improve the safety of the convoy [45]. In spite of its advantages, the security implications of deploying the middle-join and exit maneuvers were not thoroughly analyzed before this investigation.

Existing solutions for join [29] and exit processes [17, 29, 49] require splitting the platoon. However, this would enable the following member of the joiner or leaver to be promoted to a leading position (without any leader election mechanism [19, 36]). In the presence of malicious insiders, any member could be elevated to leader, thus, enabled to misbehave in various ways, e.g., preventing other vehicles from joining the platoon, or destabilizing the platoon by disseminating erroneous information. Furthermore, different schemes for exit maneuvers are considered [9, 68], but their resilience to falsifications attacks is not explored. We improved the state-of-the-art platooning formation [6, 17] by designing and deploying middle-join and exit processes. Unlike [17, 29, 49], our middle-join and exit protocols (See Fig. 17 and Fig. 18 in Appendix B) mitigate the *'privilege escalation'* attack, as they do not split the platoon nor elevate a node to a (temporary) leader.

Several consistency and plausibility mechanisms are evaluated using a mobility data-set [38]: a misbehavior detection system can detect speed and position falsification attacks with ≈40% and ≈85% accuracy, respectively. Alternatively, one can correlate the information from the predecessor and the leader to establish trust among the platoon members [31]. These results show that such an approach can detect falsification attacks; but, fail to do so in case of a compromised/malicious leader.

Appropriate countermeasures to prevent location spoofing, secure neighborhood discovery [22, 47, 52, 54, 55, 75] and physical position verification [30], can be leveraged to cross-reference other kinematic data [16]. However, such schemes are either orthogonal and complementary, yet invaluable, for automated driving (e.g., Global Navigation Satellite System (GNSS) anti-spoofing solutions); require honest neighbors (e.g., for secure neighbor discovery), or can partially assist (e.g., neighbor position verification). Nonetheless, such schemes cannot necessarily guarantee the detection of intelligent adversarial behavior, e.g., attacks that incrementally falsify the kinematic data within the limits of neighborhood verification.

Several schemes leverage Kalman Filters to identify sensor malfunctioning and misbehavior in platooning scenarios, e.g., [35, 66, 70]. A kinematic fusion approach, leveraging a Kalman Filter to validate the incoming information and handle sensor errors was investigated [35], but such a mechanism is computationally expensive [70]. To the best of our knowledge, none of the above mentioned schemes can thwart falsification attacks during maneuvering. We evaluate the Kalman Filter effectiveness during maneuvering; our results illustrate that it cannot discern between an attack and the maneuver processes, always resulting in false positives.

A Hidden Markov Model (HMM)-based scheme to detect maneuvers was proposed [34, 65]: the model was trained to recognize overtaking and lane changing maneuvers. The schemes utilize the *Forward* and *Viterbi* algorithms [57] to categorize the observed vehicle behavior. Similarly, we leverage an HMM-based approach to deign a novel MDS: beyond recognizing vehicle behavior and platoon maneuvering, we utilize the HMM-based approach to identify misbehavior and attacks during the platoon lifetime.

Machine learning based techniques were used in the VC systems to detect vehicles exhibiting abnormal behavior. Supervised learning models were investigated in [64] while semi-supervised learning and unsupervised learning were considered in [44]. More complex neural networks models were investigated with a complete pipeline of local and global detection algorithms to classify misbehavior [39]. However, none of these works consider misbehavior detection during platoon maneuvers. A detailed analysis and comparison of machine learning algorithms for broad-scope MDS in VC systems is orthogonal to this investigation.

4 NEW ATTACKS AGAINST PLATOONING

Internal adversaries, i.e., malicious platoon members, possess valid VC cryptographic material and their messages are considered legitimate, i.e, accepted, by the other platoon members. Thus, they conduct falsification attacks, by manipulating information in the CAMs or DENMs they transmit, notably speed, acceleration, and position. Furthermore, they degrade availability to harm the platoon operations by clogging DoS and/or jamming attacks. Non-platoon members also possess valid cryptographic material as part of the VC system; they cannot affect platoon functionality, but, they can launch DoS or jamming attacks; we term those as *external to the platoon* adversaries. Adversaries are not able to successfully 'crack' cryptographic keys and forge messages impersonating platoon members. We consider the general adversary model in [53] for secure and privacy-preserving VC systems and adversarial model assumptions of the vehicular platooning in the literature [18, 24, 35, 68, 69].

Novel aspects for VC platooning systems: We extend the adversarial model for vehicular platooning to (i) include collaboration among (multiple) internal and/or external adversaries; for example, an internal adversary could perform a data falsification attack and collude with an external adversary that mounts a targeted jamming attack. Moreover, we introduce (ii) the notion of rationality for platoon adversaries: an attacker could opt out of the platoon after mounting an attack, or choose faulty values that would destabilize vehicles downstream. This reduces the risk for the attacker to be itself part of a vehicle collision. This new dimension facilitates evaluating the impact of attacks on the platoon formation and can serve as a guide for adversarial choice of attack strategy (See Table 5)[1].

An internal adversary can infer platoon-specific information, e.g., the communication range and the role of the platoon members, towards maximizing the impact of its attack. Additionally, by observing the behavior of the platoon during the join and exit

[1]It is possible, of course, that internal adversaries are malware-infected vehicles that serve to mount destabilizing or destructive attacks anyway.

maneuvers, for a specific controller, it can anticipate the time of completion of future join or leave processes. This, enables an adversary to find an optimal time and position for an attack scenario, e.g., a position falsification attack when the platoon accelerates. Note that such an adversary complies with the policies determined by the leader and controllers, e.g., only manipulating the kinematic values within the plausible ranges, without interfering with the platoon protocols (thus being harder to detect).

We provide a number of different attack procedures based on this adversarial model. The attacks range from simply and statically falsifying a kinematic value (in CAMs), to launching jamming attacks, to more intelligent attacks that dynamically change all the kinematic properties with carefully selected falsified information. The latter, termed *combined* attack, affects the operation of all controllers and makes it harder for misbehavior detection mechanisms to 'flag' it as such. The adversary modifies a kinematic property (e.g. position), in each time-step, with a predetermined (system specific parameter) small value relative to its own mobility properties. Based on the position change, it then falsifies the acceleration and speed to make the former appear genuine. Alternatively, the attacker can perform a *gradual attack*, e.g., gradual acceleration attack, by incrementally increasing the falsified value.

We introduce a *novel* collusion attack by an internal and an external adversary. An external entity with a transmitter, possibly a drone hovering above the platoon or a vehicle in an adjacent lane, interrupts the communication of the targeted vehicle; while the internal adversary disseminates falsified kinematic information. Furthermore, we introduce two *novel* attack scenarios during the maneuvering processes. In both maneuvers, the attacker starts its message falsification attack when the distance to its follower (jfollower) is at its maximum. The reason is two-fold: first, the increased distance permits higher speeds to be achieved under (relatively) positive falsified values[2]; second, the joiner is affected and needs to adjust its speed just before entering the formation. This results in deviations from the nominal intra-platoon distances when completing the maneuver; moreover, the controller would still be affected by the falsified information endangering the platoon. For an exit, the attacker initiates the misbehavior once the process finishes. At this point, the new follower of the attacker accelerates to close the gap, leading to a higher collision impact.

5 PLATOON DEFENSIVE MECHANISMS

Secure Maneuvering Protocols: The protocols we design not only adhere to all the cryptographic primitives required by the state-of-the-art VCs, but also manage to thwart the *privilege escalation* attack. Due to space limitations, we detail all the secure messaging exchange for platoon maneuvering in Appendix B (Please see Fig. 17 and Fig. 18). Next, we elaborate on the proposed misbehavior detection and mitigation scheme.

Misbehavior Detection: We leverage a Gaussian Mixture Model Hidden Markov Model (GMMHMM) [57] to detect falsification attacks during maneuvering. Fig. 3 illustrates the *training phase* of the proposed MDS. The first step is to isolate the vehicle observations that correspond to one of the three maneuvering processes, i.e., static (no maneuver, only forward movement and velocity adjustments), middle-join and exit. To gather these, we run platooning simulations with and without a maneuvering process; one simulation per controller and speed combination. The observations ($o_t = (rd_{i,j}, rv_{i,j})$) consist of the relative distance and speed between a vehicle, i, and its predecessor, j, at time t.

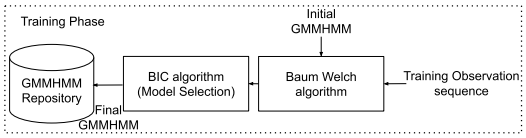


Figure 3: Misbehavior Detection Scheme (MDS) training.

As model parameters $\lambda = (A, B, \pi)$ (Sec. 2) are not known a-priori, we use the Baum-Welch algorithm [57], an iterative procedure that finds local maximum likelihoods given an observation set, to find the best initial parameters. However, the main goal of training is to find an optimal number of components in a GMMHMM model: choosing a high number results in over-fitted models and degraded performance. Thus, we use the Bayesian Information Criterion (BIC) [62] algorithm to choose an optimal number for our models, i.e., we select the best model representation of the maneuver (the numbers are given in Table 4). Each vehicle is then provided with the trained models, and the observation set that produced it.

During the *inference phase*, the MDS in each vehicle is provided with a number of recent observations ($rd_{i,j}, rv_{i,j}$), corresponding to a predefined (system parameter) interval, termed *sliding window*. These observations are then used to determine the matching behavioral maneuvering model. Because our detection mechanism operates on the observations produced by the controllers, it can better alleviate deviations produced by road traction differences and environmental effects; the controller will accelerate or decelerate in order to maintain the same distance between the vehicles.

We consider four different misbehavior detection approaches. The first two require, first, the detection of a maneuver, and then an inference on the presence of an attack. The latter two can detect an attack directly (and the maneuver implicitly). In order to avoid false attack attributions, from small deviations across the observations, e.g., from sensor errors, we apply a parameterized threshold.

Maneuver Recognition before Attack Detection: We use the *Viterbi* algorithm, a dynamic programming algorithm that produces the most probable hidden state chain from the given observations. By utilizing the four available observation sets (the current and the three modeling sets), we compare the similarity of the chains, pair-wise, to determine the presence of a maneuver. Alternatively, we use the *Forward* algorithm to compute the log likelihood of the current observation against the maneuvering models. The model that produces the biggest log likelihood corresponds to the most likely maneuver. At this point, we have only determined (or not) the presence of a maneuver. To decide if an attack takes place, we score, using the Forward algorithm, the current observation set and the model-producing set against the picked model. The score difference is then measured using the Hellinger distance; this, measures the distance between two probability distributions.

[2]Positive/negative values correspond to absolute values while *relative* positive/negative values refer to changes from the original value.

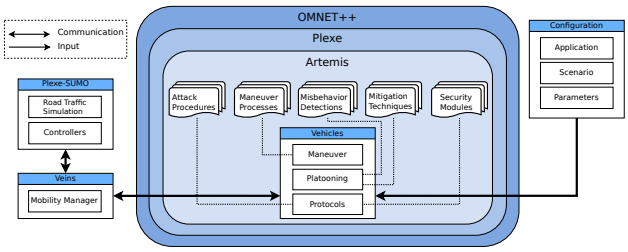


Figure 4: An Overview of the Artemis Framework.

Only Attack Detection: We use the Forward algorithm on all our observation sets in a pair-wise fashion, i.e., the current data and each of the observation sets are measured against the corresponding model. Then, we measure the Hellinger distance for each of these combinations; the minimum distance produced gives us both the most likely model and a measurable deviation between the two datasets. Alternatively, instead of applying the Hellinger distance we measure the Log Likelihood Ratio (LLR) between the log likelihood pairs. Because the observation set that produced the model can be regarded as the maximum likelihood estimate, the ratio gives us the relative plausibility of the current observation against the model-producing observations. By subtracting the log likelihoods produced for each pair L(A) (current set) and L(B) (modeling set) we obtain the likelihood ratio: $\log \frac{L(A)}{L(B)} = \log L(A) - \log L(B)$

Mitigation Techniques: Detecting an attack is the first step in safeguarding the platooning application. When a misbehavior is detected, a mitigation technique can be employed to avoid its effects: degrading from CACC to ACC [31, 35] and/or gradually increasing the intra-platoon distances [74]. These mitigation techniques are considered in our framework set-up in order to preserve the safety of the platoon. However, detecting an attack does not guarantee that its effect will be mitigated; sophisticated attacks can still induce instability and/or crashes (See Appendix C).

6 SIMULATION FRAMEWORK

We create Artemis, a platooning security assessment framework, an extensible tool shown in Fig. 4, for the systematic evaluation of cyberphysical attacks, including notably their physical effects and the effectiveness of the proposed defenses. We use OMNET++ [5], Plexe [6, 35], an extension of Veins [50], and SUMO [20] as the building blocks. This ensemble provides a comprehensive realistic simulation environment: a complex cyber-physical setting, covering vehicular mobility and perception in VC systems, network connectivity (802.11p [4]) and platooning automation. We expand Plexe and SUMO by implementing new maneuvers (middle-join and leave processes), so that any position of the platoon can be used; a step for more complex maneuvering scenarios, e.g., the middle-join for a two-platoon merge. We extend all the existing controllers (PATH, Ploeg, Consensus and Flatbed) with the extended set of maneuvers, including, naturally, join at the platoon tail. We implement all attacks described in Sec. 4. The Artemis framework supports positioning the attacker(s) at any position in the formation and it allows the collaboration of multiple (internal or external) adversaries to perform different type of attacks.

Table 2: Simulation Parameters for the Experiments.

Parameters	Value	Parameters	Value
Beacon interval	0.1s	Controller	PATH, Ploeg, Consensus, Flatbed
Carrier frequency	5.89 GHz	Spacing	5m, 0.5s, 0.8s, 5m
Physical layer bit-rate	6 Mbps	Leader speed	50, 80, 100, 150 kmph
Area size	5 KM × 50 M 4 lanes	Sensors	$\epsilon_p^{V2V} = 1m$, $\epsilon_s^{V2V} = 0.1m/s$, $\epsilon_a^{V2V} = 0.01m/s^2$, $\epsilon_p^{RAD} = 0.1m$, $\epsilon_s^{RAD} = 0.1m/s$
Number of vehicles	6 - 7 (Join)	Falsification steps	$Position_{step} = 2.5m$ $Speed_{step} = 0.5m/s$ $Acceleration_{step} = 0.05m/s^2$
Propagation delays	Randomized	Kalman Filter (KF)	$AverageWindow = 10$ values $Tolerance = 10$ detections $distance_{threshold}^{KF} = 0.33$ $distance_{threshold}^{RAD} = 0.25$ $acceleration_{factor} = 0.05$
Vehicle TX Range	600m	Leader speed pattern	Sinusoidal
Jammer TX Range	50m	Number of executions	10 times
TX power	100mW	Duration of simulation	120s
Thermal Noise	-95dBm	Warm-up period	30s
Sensitivity	-94dBm	Mitigation Headway	2s
Sliding window	1s-9(1)s	Vehicle length	4 m

Table 3: Attacker Configuration Parameters.

Configuration	Attack Values	Attack Position	Maneuver
PosAttack	{3, 5, 7, 9, 11} m	[0, 2]	[no, Join, Exit]
SpeedAttack	{-50, 0, 50, 100, 150} km/h	[0, 2]	[no, Join, Exit]
AccAttack	{-30, -10, 0, 10, 30} m/s^2	[0, 2]	[no, Join, Exit]
GradualPosAttack	[-10,40] m	[0, 2]	[no, Join, Exit]
GradualSpeedAttack	[-10,17] m/s	[0, 2]	[no, Join, Exit]
GradualAccAttack	[-10,10] m/s^2	[0, 2]	[no, Join, Exit]
CombinedPosAttack	[-10,10] m	[0, 2]	[no, Join, Exit]
CombinedSpeedAttack	[-10,10] m/s	[0, 2]	[no, Join, Exit]
CombinedAccAttack	[-10,10] m/s^2	[0, 2]	[no, Join, Exit]
ColludingAttack	40 s	[2 and 4]	[no]

We implement a gamut of countermeasures, including notably the Kalman Filter-based data fusion MDS and our proposal. Specifically, (i) a Kalman Filter [35] detection mechanism to identify falsification attacks, implemented using the Armadillo math library [8, 59, 60] and (ii) our machine-learning based approach based on GMMHMM (using the hmmlearn and sklearn python libraries). Artemis allows the implementation and integration of other defensive mechanisms, which function either offline, as forensic mechanisms, or online as an MDS, through a C++/Python API that connects the MDS module with the main simulation software.

Last but not least, Artemis supports a range of sensors and allows imposing sensor errors (through the use of configuration parameters) to approximate a real-world environment. It is also equipped with a security module for basic security and privacy requirements, e.g., message signature generation and validation. All options can be utilized by one or more vehicles during the simulation, depending on the intended outcome of the experiment; any combination of the attack procedures, maneuvering processes, misbehavior detection and mitigation techniques is permitted. To systematically evaluate the performance of the platooning applications under various attack scenarios, we utilize a set of metrics to assess the transportation safety and the performance of the CACC controllers.

Experimental Setup: Table 2 shows the simulation parameters for the experiments. Intra-platoon distances are set to 5 meters for the CVS policy (PATH and Flatbed), and the default headway values for Ploeg and Consensus controllers, 0.5 and 0.8 seconds respectively. The platoon speeds were chosen similar to real-world

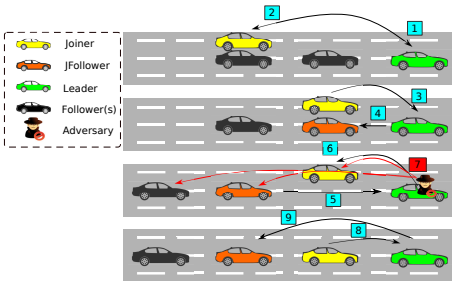


Figure 5: A Leader Falsification Attack during a Middle-Join.

scenarios, ranging from 50 km/h to 150 km/h. Similar to [35], we incorporate sensor uncertainties based on real-world sensor capabilities; errors are uniformly random for all sensors (the radar, the Global Positioning System (GPS) receiver, and the wheel spin sensor). We experimented with several sliding window values, ranging from 1 second to 9 seconds; for the detect-maneuver-then-attack approaches, the sliding window is 9 seconds when determining the maneuver, but only the last second is used to detect an attack (noted as 9(1)).

The platoon consists of six identical cars moving on a straight road with four lanes, and their V2X communication is over IEEE 802.11p [4]. During a join, a new vehicle approaches the platoon from an adjacent lane. Vehicles already possess the necessary credentials for interacting with the leader, i.e., initiating the join and leave processes. We model propagation delays in the physical, data-link, and network layers. Each experiment uses a different seed to randomize the delays and it is run ten times. We employ a 30s warm-up period with the ACC fallback headway at 2s [63].

Table 3 summarizes the parameters for the implemented attacks: the falsified values, the start of the colluding attack, and the position(s) of the attacker(s). Each attack is evaluated in three configurations: (i) a static platoon, i.e., no maneuver, just forward movement and velocity adjustments, (ii) during a middle-join maneuver, and (iii) during an exit maneuver. The attack values vary depending on the scenario, e.g., for *speed falsification* attack, the values range from static -50 to 150 km/h; for *gradual speed falsification* attack, the adversarial values fall within the range of [-10, 40] m/s. In *combined falsification attacks*, the range of falsified values ([-10,10]) is smaller than those for gradual ones: large changes in position within a short time frame, e.g., 100 ms, would require changes in speed and acceleration beyond the plausible ranges, thus, being flagged as malicious. The attacking positions are relative to the platoon structure: 0 (for the leader) or 2 (for the third platoon member). For colluding attacks, the adversary conducts an *intelligent* attack: a gradual or a combined falsification (positioned at 2), in conjunction with a *targeted jamming attack* (victim positioned at 4 cannot receive messages).

For static platooning, the attack starts when the platoon begins to accelerate, as this has the highest collision impact [17] (See Appendix E). For a meaningful comparison among controllers, we perform the attacks at a specific step during the maneuvering (step 17.20, for the join process, as shown in Fig. 5, and step 18.10, for the leave process). We also chose the (follower) attacker to be positioned immediately ahead of either the joiner or the leaver, i.e., an ideal placement to induce the highest impact/harm. The reasoning behind this position and the ability of an attacker to predict the maneuvering process is presented in Sec. 4. Table 4 summarizes the number of components for each maneuver based on speed and controller. These values show the optimal number of components to maximize the MDS performance as described in Sec. 5.

Table 4: The Optimal Number of GMMHMM Components.

Controller	Speed (km/h)	No Maneuver	Middle Join	Exit
Path	(50,80,100,150)	(9,11,9,11)	(13,13,13,13)	(11,9,9,9)
Ploeg	(50,80,100,150)	(11,9,9,9)	(9,9,7,9)	(11,11,9,11)
Consensus	(50,80,100,150)	(7,9,11,7)	(8,9,9,9)	(11,11,11,11)
Flatbed	(50,80,100,150)	(9,6,7,9)	(17,19,19,17)	(11,11,11,11)

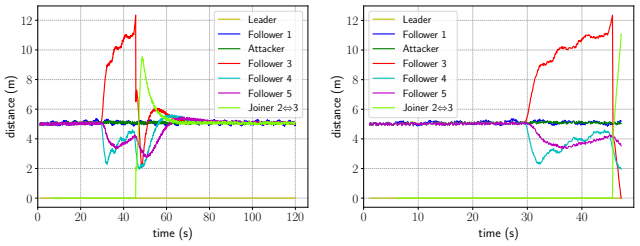
Metrics: To evaluate the *string stability* of a platoon, we measure the spacing errors induced by the attacks; to gauge the *collision impact*, i.e., the level of crash severity, we measure the ΔV (*speed difference*) of the vehicles involved. For the controllers, we consider *resiliency* in terms of them being in a *stable*, *unstable* or *crash* condition. A controller is stable if its behavior is not affected by an attack and unstable if it is affected, but without resulting into a crash. For experiments leading to a crash, we quantify the involvement of an attacker: being part of a crash is undesirable as it is harmful to the attacker. The controller comparison is given in Appendix D.

The metrics to evaluate the MDS are *recall*, *precision*, *accuracy*, F_1 *score*, and *Receiver Operating Characteristic (ROC)* curves. These metrics provide additional details on the results of the classifier algorithms and they are based on the True Positive (TP), False Positive (FP), True Negative (TN), and False Negative (FN) metrics. *Recall* is the proportion of the correctly identified messages (TP) over the correctly identified messages and messages falsely flagged as benign (TP+FN). *Precision* is the proportion of correctly identified messages (TP) over all identified messages (TP+FP). F_1 *score* provides the harmonic mean of *recall* and *precision*. *ROC*, the proportion of correctly detected messages (TP) over incorrectly identified messages (FP), is presented in Appendix F. To measure the overhead of identifying a maneuver and detecting an attack, we test three different systems using a Python time library [11] to count specifically the CPU time of the MDS process. These results can be found in Appendix G.

7 ATTACK ANALYSIS RESULTS

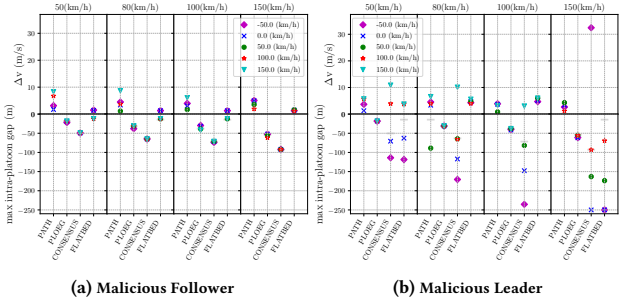
Attacks During the Middle-Join Process: During the middle-join, the joiner needs to maintain its relative distance to the entry position; it relies on information by its predecessor, an attacker in this scenario. Fig. 6.a shows that a falsified positive acceleration has a minor effect (momentary instability) on the formation with Flatbed. In contrast, Fig. 6.b shows that a falsified negative acceleration results in a collision. The affected vehicle continues to decelerate even after changing its controller to the platoon appropriate controller, in this case Flatbed. This is important because Flatbed is not affected by acceleration attacks.

Fig. 7 explores the combined position falsification attack during a middle-join. We consider the *collision impact* metric, i.e., the severity of a crash (measured with ΔV), shown on the top-half of the



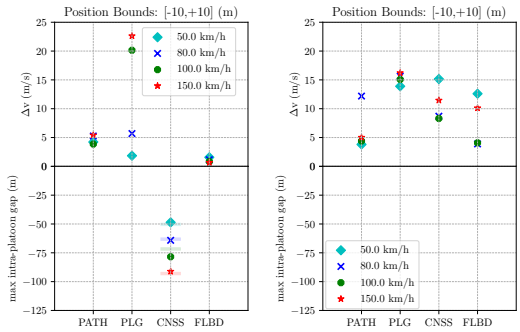
(a) Acceleration: +10 (m/s^2) (b) Acceleration: -10 (m/s^2)

Figure 6: Acceleration Falsification on Flatbed During Join.



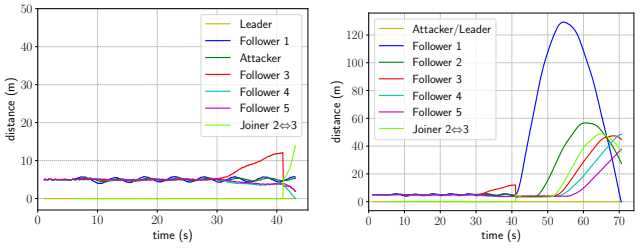
(a) Malicious Follower (b) Malicious Leader

Figure 9: Attack Impact: Speed Falsification During Join.



(a) Malicious Follower (b) Malicious Leader

Figure 7: Attack Impact: Combined Position During Join.



(a) Malicious Follower (b) Malicious Leader

Figure 8: Gradual Acceleration on Path During Join.

figure, and the *string stability* metric, i.e., the spacing error induced by an attack, shown on the bottom-half of the figure. The colored bars in each column correspond to the nominal intra-platoon gaps. Fig. 7.a shows the combined position falsification attack during the middle-join when the follower misbehaves: all controllers are affected, suffering a collision, except Consensus that does not utilize predecessor data. Fig. 7.b illustrates the combined position falsification for a leader-attacker. All the controllers are affected, suffering collisions, with impact ranges from ≈18 to 58 km/h.

Fig. 8 presents the gradual acceleration falsification attack on the Path controller during middle-join. Fig. 8.a shows the decrease of the intra-platoon distances of the vehicles when the attack is performed by the follower; upon maneuver completion, the downstream vehicles collide without the attacker being part of the collision. Fig. 8.b shows the gradual acceleration attack by the leader: once the maneuver is completed, all the platoon members are affected but the immediate follower of the leader collides with the attacker. The gradual acceleration attack by the follower is significantly less harmful, but it does not involve the attacking vehicle.

Fig. 9 compares speed falsification: Path controller always leads to a crash; however, the crash severity is slightly higher when the attack is performed by the follower. With Flatbed, the *jfollower* crashes into the joiner when the attacker disseminates relatively negative speeds. With a misbehaving leader, the impact is amplified. For example, the attack with relatively negative falsified values at 50 km/h results in instability, while the positive values cause a collision. Consensus, though, is not affected by a malicious follower but rather a misbehaving leader: with relatively negative falsified values, the intra-platoon gaps grow from 35 meters to 230 meters, when the speed is 100 km/h. Ploeg is never affected as it does not consider the disseminated speed values from V2V communications.

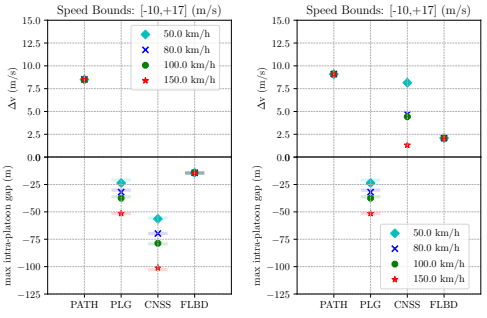
Attacks During the Exit Process: PATH is susceptible to gradual speed falsification attacks during the exit maneuver. Fig. 10.a shows that the follower of the attacker crashes into the attacking vehicle; however, the other controllers are resilient. PATH, Flatbed, and Consensus suffer from collisions against a leader-mounted gradual speed attack (Fig. 10.b). Consensus decreases the rate of acceleration to compensate for the deviation from the nominal vehicle distance; the higher the speed, the larger the intra-platoon distances, thus, the smaller speed difference for the colliding vehicles.

The combined acceleration falsification attack results in a crash when using PATH and Ploeg (Fig. 11). With PATH, the collision impact caused by the misbehaving leader is greater than the one caused by a malicious follower. Fig. 11.b shows that the attack with Flatbed results in an accident when the leader falsifies all kinematic properties; however, the crash severity is less than 5 km/h.

Colluding Attackers: Fig. 12 evaluates controllers under a gradual position falsification attack (Fig. 12.a) combined with a jamming attack (Fig. 12.b). By actively disrupting communication downstream, a collision is caused for PATH and Flatbed, but not for Consensus and Ploeg. Consensus increases intra-platoon distances if CAMs are not received within a time interval (100 ms). Ploeg estimates the acceleration of its predecessor (via its radar sensor)

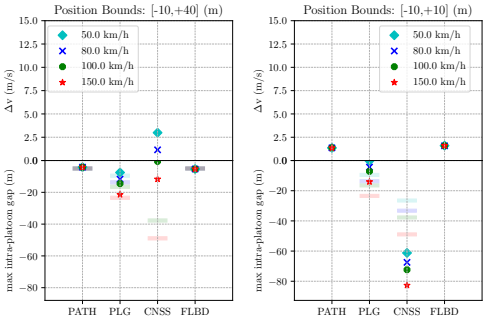
Table 5: Collision Hitmap: attacker not in collision (green); attacker in collision (red); Follower (F) and Leader (L).

Attack		No Maneuver				Join Maneuver				Exit Maneuver			
		50 (km/h)	80 (km/h)	100 (km/h)	150 (km/h)	50 (km/h)	80 (km/h)	100 (km/h)	150 (km/h)	50 (km/h)	80 (km/h)	100 (km/h)	150 (km/h)
Position (m)	PATH	-	-	-	-	-	-	-	-	-	-	-	-
	Consensus	LF {3,5,7,9,11}	LF {3,5,7,9,11}	LF {3,5,7,9,11}	LF {3,5,7,9,11}	LF {3,5,7,9,11}	LF {3,5,7,9,11}	LF {3,5,7,9,11}	LF {3,5,7,9,11}	LF {3,5,7,9,11}	LF {3,5,7,9,11}	LF {3,5,7,9,11}	LF {3,5,7,9,11}
	Flatbed	-	-	-	-	-	-	-	-	-	-	-	-
	Ploeg	-	-	-	-	-	-	-	-	-	-	-	-
Speed (km/h)	PATH	LF {-50,0}	LF {-50,0} F 50	LF {-50,0,50}	LF {-50,0,50,100}	LF {-50,0}	LF {-50,0} F 50	LF {-50,0,50}	LF {-50,0,50,100}	LF {-50,0}	LF {-50,0} F 50	LF {-50,0,50}	LF {-50,0,50,100}
	Consensus	L {100,150}	L 150	L 150	-	L {100,150}	L 150	L {0, 50}	L -50	L {100,150}	L 150	L 150	-
	Flatbed	L {100,150}	L {100,150}	L 150	-	F {-50,0}	LF {-50,0} L 50	LF {-50,0} L 50	F {-50,0,50,100}	L {100,150}	L {100,150}	L 150	-
	Ploeg	-	-	-	-	-	-	-	-	-	-	-	-
Acceleration (m/s^2)	PATH	F {-30, -10}	F {-30, -10}	F {-30, -10}	F {-30, -10}	F {-30, -10}	LF -30, F -10	LF -30, F -10	LF -30, F -10	F {-30, -10}	F {-30, -10}	F {-30, -10}	F {-30, -10}
	Consensus	-	-	-	-	-	-	-	-	-	-	-	-
	Flatbed	-	-	-	-	LF -10, F -30	F {-30, -10}	L -10, F -30	F {-30, -10,10,30}	-	-	-	-
	Ploeg	LF {-10, -30}	LF -30	LF -30	LF -30	L -10, LF -30	LF -30	LF -30	LF {10,30}	LF -30	LF -30	LF -30	L -30
Gradual Position (m)	PATH	-	-	-	-	-	-	-	-	-	-	-	-
	Consensus	LF -10/+40	LF -10/+40	-	-	LF -10/+40	LF -10/+40	L -10/+40	-	LF -10/+40	LF -10/+40	L -10/+40	-
	Flatbed	-	-	-	-	-	-	-	-	-	-	-	-
	Ploeg	-	-	-	-	-	-	-	-	-	-	-	-
Gradual Speed (m/s)	PATH	LF {-10/+17}	LF {-10/+17}	LF {-10/+17}	LF {-10/+17}	LF -10/+17	LF -10/+17	LF -10/+17	LF -10/+17	LF -10/+17	LF -10/+17	LF -10/+17	LF -10/+17
	Consensus	L -10/+17	L -10/+17	L -10/+17	L -10/+17	L -10/+17	L -10/+17	L -10/+17	L -10/+17	L -10/+17	L -10/+17	L -10/+17	L -10/+17
	Flatbed	L -10/+17	L -10/+17	L -10/+17	L -10/+17	L -10/+17	L -10/+17	L -10/+17	L -10/+17	L -10/+17	L -10/+17	L -10/+17	L -10/+17
	Ploeg	-	-	-	-	-	-	-	-	-	-	-	-
Gradual Acceleration (m/s^2)	PATH	L -10/+10	F -10/+10	F -10/+10	F -10/+10	F -10/+10	F -10/+10	F -10/+10	F -10/+10	F -10/+10	F -10/+10	F -10/+10	F -10/+10
	Consensus	-	-	-	-	-	-	-	-	-	-	-	-
	Flatbed	-	-	-	-	LF -10/+10	-	L -10/+10	F -10/+10	-	-	-	-
	Ploeg	LF -10/+10	LF -10/+10	LF -10/+10	LF -10/+10	L -10/+10	LF -10/+10	LF -10/+10	LF -10/+10	LF -10/+10	LF -10/+10	LF -10/+10	LF -10/+10
Combined Position (m)	PATH	L -10/+10	F -10/+10	F -10/+10	F -10/+10	F -10/+10	F -10/+10	F -10/+10	F -10/+10	F -10/+10	F -10/+10	F -10/+10	F -10/+10
	Consensus	L -10/+10	L -10/+10	L -10/+10	L -10/+10	L -10/+10	L -10/+10	L -10/+10	L -10/+10	L -10/+10	L -10/+10	L -10/+10	L -10/+10
	Flatbed	F -10/+10	F -10/+10	F -10/+10	F -10/+10	LF -10/+10	LF -10/+10	L -10/+10	LF -10/+10	LF -10/+10	LF -10/+10	LF -10/+10	LF -10/+10
	Ploeg	L -10/+10	LF -10/+10	LF -10/+10	LF -10/+10	F -10/+10	F -10/+10	LF -10/+10	LF -10/+10	LF -10/+10	LF -10/+10	LF -10/+10	LF -10/+10
Combined Speed (m/s)	PATH	F -10/+10	F -10/+10	F -10/+10	F -10/+10	F -10/+10	LF -10/+10	L -10/+10	LF -10/+10	F -10/+10	F -10/+10	F -10/+10	F -10/+10
	Consensus	L -10/+10	L -10/+10	-	-	L -10/+10	L -10/+10	L -10/+10	-	L -10/+10	L -10/+10	-	-
	Flatbed	L -10/+10	L -10/+10	L -10/+10	L -10/+10	LF -10/+10	L -10/+10	L -10/+10	L -10/+10	L -10/+10	L -10/+10	L -10/+10	L -10/+10
	Ploeg	LF -10/+10	LF -10/+10	L -10/+10	LF -10/+10	LF -10/+10	LF -10/+10	LF -10/+10	LF -10/+10	LF -10/+10	LF -10/+10	L -10/+10	LF -10/+10
Combined Acceleration (m/s^2)	PATH	F -10/+10	F -10/+10	F -10/+10	F -10/+10	F -10/+10	F -10/+10	F -10/+10	F -10/+10	F -10/+10	F -10/+10	F -10/+10	F -10/+10
	Consensus	-	-	-	-	-	-	-	-	-	-	-	-
	Flatbed	L -10/+10	L -10/+10	L -10/+10	L -10/+10	L -10/+10	L -10/+10	L -10/+10	L -10/+10	L -10/+10	L -10/+10	L -10/+10	L -10/+10
	Ploeg	LF -10/+10	LF -10/+10	LF -10/+10	LF -10/+10	L -10/+10	LF -10/+10	LF -10/+10	LF -10/+10	LF -10/+10	LF -10/+10	LF -10/+10	LF -10/+10



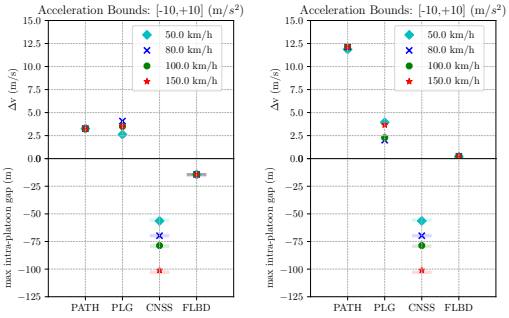
(a) Malicious Follower (b) Malicious Leader

Figure 10: Attack Impact: Gradual Speed During Exit.



(a) Follower Attacks (b) Follower/Drone Attacks

Figure 12: Attack Impact: Gradual Position with Colluder.

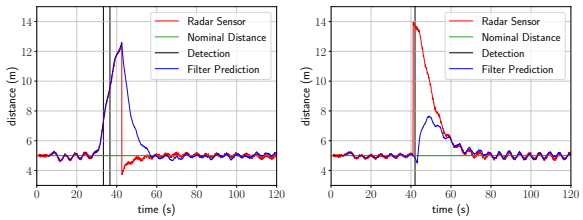


(a) Malicious Follower (b) Malicious Leader

Figure 11: Attack Impact: Combined Acceleration on Exit.

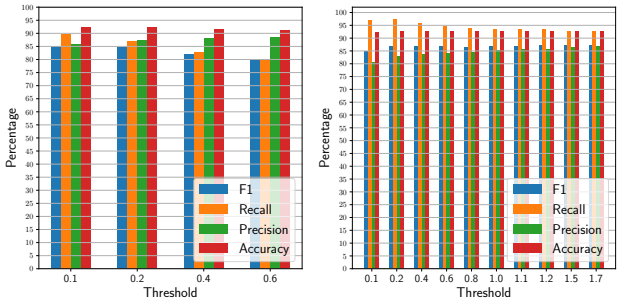
if the CAMs are not received on time, resulting only in a destabilize platoon. Thus, the impact of collusion attacks depends on the controller characteristics.

Attacker Hitmap: Table 5 summarizes all the attack scenarios that resulted in a collision from the perspective of a rational adversary. Attacks involving the adversarial vehicle are marked as red; when the attacker causes a crash without being part of the collision, we mark it as green (when both happen, we prioritize the safety of the attacker).F denotes attacks mounted by a follower and L those mounted by the leader. A simple scenario that illustrates this is presented in Appendix A. PATH is susceptible to speed and acceleration falsification attacks, when the falsified speed and acceleration are lower than the victim's values. For PATH, a rational adversary would choose relatively negative speed and acceleration values. Consensus appears to be an inappropriate target for a rational attacker: without a maneuver, all the accidents involve the misbehaving vehicle (marked as red). An adversary positioned as follower and targeting Flatbed would avoid attacking during the exit maneuver: the attack never results into a collision, except for the combined position scenario (red cell). For combined speed falsification attacks against Ploeg, a rational adversary would opt



(a) During Middle-Join (b) During Exit

Figure 13: Maneuver Detection with a Kalman Filter



(a) Different Thresholds: Hellinger (b) Different Thresholds: LLR

Figure 14: Detection Rates: 1 second window

in attacking the formation at lower speeds (50 km/h and 80 km/h) regardless of its position or the undertaken maneuver.

8 MISBEHAVIOR DETECTION ANALYSIS

Kinematic Fusion and Kalman Filter: We incorporated a state-of-the-art MDS [35] that leverages kinematic data fusion and a Kalman Filter to identify misbehavior. The scheme relies on a rolling average[3] to calculate the plausibility of the incoming (kinematic) data to detect deviations. The Kalman Filter normalizes sensor errors, e.g., from the GPS sensor, in order to improve the accuracy of the detection model. We find that even though such an approach detects attacks during static platooning [35], it fails to detect attacks during maneuvering.

Fig. 13.a and Fig. 13.b show that the scheme fails to distinguish between attacks and benign middle-join and exit maneuvers (falsely marked as misbehavior). This is due to the extreme divergence of vehicle behavior from '*normality*' during the maneuvers. During a middle-join (Fig. 13.a), the distance of the *jfollower* from its predecessor (red line) deviates from its nominal intra-platoon distance (green line). The scheme estimates the increase in distance (blue line) and flags the deviation as a misbehavior (black lines). In fact, multiple vehicles flag their predecessor as malicious: the first detection happens by the *jfollower*, while the second one by its follower. Similarly, Fig. 13.b shows that the system falsely flags the exit process as a malicious behavior even without any attacks. This mis-identification is present under all maneuvering scenarios.

[3]Computed as the average of the last k values over a series of n values ($k < n$).

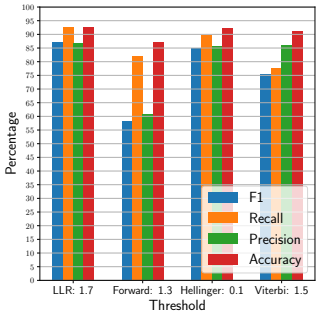


Figure 15: Comparison Between Best F1 Scores

GMMHMM-based Misbehavior Detection: Our MDS is not only capable of understanding if a maneuver takes place, but it can also detect falsification attacks during maneuvering. Fig. 14 shows the performance of the attack-detection-only approaches (Hellinger and LLR) for a sliding window of 1 second. Fig. 14.a shows the precision metric increases with higher detection thresholds. However, this reduces the recall. A higher threshold results in lower FP and higher FN . Similarly, Fig. 14.b shows the trend of improved precision as we increase the threshold; the improvement plateaus when we reach a threshold/ratio of 1.7.

In order to compare all four approaches, and in the interest of space, we show in Fig. 15 the best results produced by each. We identify as best the threshold and window combination that produces the higher F_1 metric. As the approaches differ, the detection thresholds also differ. Nonetheless, we observe that a pure Forward algorithm approach (both for maneuver detection and attack recognition) produces poor results. In contrast, the Forward algorithm, in conjunction with the LLR approach, produces the best F_1 score; the margin between this approach and Hellinger is 3%. The feasibility in terms of computational overhead, measuring MDS latencies, is given in Appendix G.

9 CONCLUSION AND FUTURE WORK

Paving the way for the deployment of secure platooning systems requires a systematic investigation of attacks and detection mechanisms. Our framework, *Artemis*, offers new capabilities in performing middle-join and leave maneuvers for all control algorithm. We investigate the attack impact and how adversaries could harm the platoon based on their intelligence and rationality. We show that unlike a kinematic data fusion MDS, leveraging a Kalman Filter, our novel MDS scheme is capable of detecting such attacks. As future work, we will expand Artemis to include merge and split maneuvers and an automatic configuration tuning, based on attack impact, towards an improved misbehavior detection.

ACKNOWLEDGMENTS

Work supported by the Swedish Foundation for Strategic Research (SSF) SURPRISE project and the KAW Academy Fellowship Trustworthy IoT project.

REFERENCES

[1] 2014. Nexcom VTC 6201. https://www.nexcom.com/fildata/getpdf/744ea2f3-dcfa-4ea4-bbe2-9685917ebe67

[2] 2014. Taxonomy and Definitions for Terms Related to On-Road Motor Vehicle Automated Driving Systems. https://www.sae.org/standards/content/j3016_201401/

[3] 2016. IEEE Standard for Wireless Access in Vehicular Environments - Security Services for Applications and Management Messages. *IEEE Std 1609.2* (Mar. 2016).

[4] 2016. IEEE Standard for Wireless Access in Vehicular Environments (WAVE) - Networking Services. *IEEE Vehicular Technology Society* (Jan. 2016).

[5] 2017. OMNeT++. https://www.omnetpp.org/.

[6] 2020. Plexe: The Platooning Extension for Veins. http://plexe.car2x.org/.

[7] 2020. Sweden4Platooning Project. https://www.trafikverket.se/contentassets/e9dec2f430b94466a8aa4572966aeb39/publik-rapport_tsaf_20200130.pdf

[8] 2021. C++ library for linear algebra & scientific computing. arma.sourceforge.net

[9] 2021. The H2020 Project ENSEMBLE: Platooning usecases. https://platooningensemble.eu/storage/uploads/documents/2021/03/24/ENSEMBLE-D2.2_V1-Platooning-use-cases,-scenario-definition-and-platooning-levels_FINAL.pdf

[10] 2021. The H2020 Project ENSEMBLE: Requirements. https://platooningensemble.eu/storage/uploads/documents/2021/03/24/ENSEMBLE-Deliverable-2.1-StateOfTheArt_EUProjects_Final.pdf

[11] 2021. The Python Standard Library: Generic Operating System Services. https://docs.python.org/3/library/time.html

[12] Ahmed Abdo, Sakib Md Bin Malek, Zhiyun Qian, Qi Zhu, Matthew Barth, and Nael Abu-Ghazaleh. 2019. Application Level Attacks on Connected Vehicle Protocols. In *Symposium on RAID*. 459–471.

[13] Assad Al Alam, Ather Gattami, and Karl Henrik Johansson. 2010. An Experimental Study on the Fuel Reduction Potential of Heavy Duty Vehicle Platooning. In *IEEE on ITS*. Funchal, Portugal, 306–311.

[14] A. Ali, G. Garcia, and P. Martinet. 2015. The Flatbed Platoon Towing Model for Safe and Dense Platooning on Highways. *IEEE ITS Magazine* 7, 1 (Jan. 2015), 58–68.

[15] Amir Alipour-Fanid, Monireh Dabaghchian, Hengrun Zhang, and Kai Zeng. 2017. String Stability Analysis of Cooperative Adaptive Cruise Control under Jamming Attacks. In *IEEE HASE*. Singapore, Singapore.

[16] Faris Alotibi and Mai Abdelhakim. 2020. Anomaly Detection for Cooperative Adaptive Cruise Control in Autonomous Vehicles Using Statistical Learning and Kinematic Model. *IEEE TITS* (Apr. 2020).

[17] Mani Amoozadeh, Hui Deng, Chen-Nee Chuah, H Michael Zhang, and Dipak Ghosal. 2015. Platoon Management with Cooperative Adaptive Cruise Control Enabled by VANET. *Veh. Comm.* 2, 2 (Apr. 2015), 110–123.

[18] M. Amoozadeh, A. Raghuramu, C. Chuah, D. Ghosal, H. M. Zhang, J. Rowe, and K. Levitt. 2015. Security Vulnerabilities of Connected Vehicle Streams and Their Impact on Cooperative Driving. *IEEE Comm. Mag.* 53, 6 (Jun. 2015).

[19] Jakob Axelsson et al. 2020. Truck Platooning Business Case Analysis.

[20] Michael Behrisch, Laura Bieker, Jakob Erdmann, and Daniel Krajzewicz. 2011. SUMO - Simulation of Urban MObility: An Overview. In *The International Conference on Advances in System Simulation*. Barcelona, Spain.

[21] B. Besselink and K. H. Johansson. 2017. String Stability and a Delay-Bsed Spacing Policy for Vehicle Platoons Subject to Disturbances. *CoRR* abs/1702.01031 (Feb. 2017).

[22] S. Brands and D. Chaum. 1993. Distance-Bounding Protocols. In *Workshop on the Theory and Application of Cryptographic Techniques*. Berlin, Heidelberg, 344–359.

[23] Fred Browand, John McArthur, and Charles Radovich. 2004. Fuel Saving Achieved in the Field Test of Two Tandem Trucks. *Institute of Transportation Studies, UC Berkeley* (Jan. 2004).

[24] Yuri Gil Dantas, Vivek Nigam, and Carolyn Talcott. 2020. A Formal Security Assessment Framework for Cooperative Adaptive Cruise Control. In *IEEE VNC*. NY, USA.

[25] T. Dao, C. M. Clark, and J. P. Huissoon. 2007. Optimized Lane Assignment Using Inter-Vehicle Communication. In *IEEE Intelligent Vehicles Symposium*. 1217–1222.

[26] ETSI. 2016. Intelligent Transport Systems (ITS); Security; ITS Communications Security Architecture and Security Management.

[27] ETSI-TR-103-298. 2019. *Intelligent Transport Systems (ITS); Platooning; Pre-standardisation Study.* Technical Report. ETSI.

[28] ETSI-TR-103-460. 2020. *Intelligent Transport Systems; Security; Pre-standardisation Study on Misbehaviour Detection - Release 2.* Technical Report. ETSI.

[29] Nigar Anjum Fida, Naveed Ahmad, Yue Cao, Mian Ahmad Jan, and Gauhar Ali. 2021. An improved multiple manoeuver management protocol for platoon mobility in vehicular ad hoc networks. *IET Intelligent Transport Systems* (May 2021).

[30] Marco Fiore et al. 2013. Discovery and Verification of Neighbor Positions in Mobile Ad Hoc Networks. *IEEE TMC* 12, 2 (Feb. 2013), 289–303.

[31] Keno Garlichs et al. 2019. TriP: Misbehavior Detection for Dynamic Platoons using Trust. In *IEEE ITSC*. Auckland, New Zealand, 455–460.

[32] Amir Ghiasi et al. 2017. A Mixed Traffic Capacity Analysis and Lane Management Model for Connected Automated Vehicles: A Markov Chain Method. *Transportation Research Part B: Methodological* 106 (Dec. 2017), 266–292.

[33] Randolph Hall and Chinan Chin. 2005. Vehicle Sorting for Platoon Formation: Impacts on Highway Entry and Throughput. *Transportation Research Part C: Emerging Technologies* 13, 5 (2005), 405–420.

[34] Haijing Hou, Lisheng Jin, Qingning Niu, Yuqin Sun, and Meng Lu. 2011. Driver Intention Recognition Method Using Continuous Hidden Markov Model. *International Journal of Computational Intelligence Systems* 4, 3 (2011), 386–393.

[35] Marco Iorio, Fulvio Risso, Riccardo Sisto, Alberto Buttiglieri, and Massimo Reineri. 2019. Detecting Injection Attacks on Cooperative Adaptive Cruise Control. In *IEEE VNC*. Los Angeles, CA, USA, 1–8.

[36] Yongxin Ji et al. 2020. A Blockchain-Based Vehicle Platoon Leader Updating Scheme. In *IEEE International Conference on Communications (ICC)*. 1–6.

[37] Dongyao Jia et al. 2015. A Survey on Platoon-Based Vehicular Cyber-Physical Systems. *IEEE Communications Surveys & Tutorials* 18, 1 (Mar. 2015), 263–284.

[38] J. Kamel et al. 2020. VeReMi Extension: A Dataset for Comparable Evaluation of Misbehavior Detection in VANETs. In *IEEE ICC*. Dublin, Ireland.

[39] Joseph Kamel, Mohammad Raashid Ansari, Jonathan Petit, Arnaud Kaiser, Ines Ben Jemaa, and Pascal Urien. 2020. Simulation Framework for Misbehavior Detection in Vehicular Networks. *IEEE Transactions on Vehicular Technology* 69, 6 (April 2020), 6631–6643.

[40] Kanwaldeep Kaur and Giselle Rampersad. 2018. Trust in driverless cars: Investigating key factors influencing the adoption of driverless cars. *Journal of Engineering and Technology Management* 48 (Apr. 2018), 87–96.

[41] M. Khodaei et al. 2018. SECMACE: Scalable and Robust Identity and Credential Management Infrastructure in Vehicular Communication Systems. *IEEE TITS* 19, 5 (May 2018), 1430–1444.

[42] Michael P. Lammert et al. 2014. Effect of Platooning on Fuel Consumption of Class 8 Vehicles Over a Range of Speeds, Following Distances, and Mass. *SAE Int. J. Commer. Veh.* 7 (Sep. 2014), 626–639.

[43] Xiao-Yun Lu and Steven E. Shladover. 2014. *Automated Truck Platoon Control and Field Test.* Springer International Publishing, Cham, 247–261.

[44] Matthias Matousek, Mahmoud Yassin, Ala'a Al-Momani, Rens van der Heijden, and Frank Kargl. 2018. Robust Detection of Anomalous Driving Behavior. In *2018 IEEE 87th Vehicular Technology Conference (VTC Spring)*. 1–5. https://doi.org/10.1109/VTCSpring.2018.8417777

[45] A. Mihály et al. 2012. Control of platoons containing diverse vehicles with the consideration of delays and disturbances. *Periodica Polytechnica* 40, 1 (Oct. 2012).

[46] V. Milanés et al. 2014. Cooperative Adaptive Cruise Control in Real Traffic Situations. *IEEE Transactions on ITS* 15, 1 (Feb. 2014), 296–305.

[47] Sashank Narain et al. 2019. Security of GPS/INS Based On-road Location Tracking Systems. In *IEEE Symposium on Security and Privacy (SP)*. CA, USA, 587–601.

[48] G. J. L. Naus et al. 2010. String-Stable CACC Design and Experimental Validation: A Frequency-Domain Approach. *IEEE Transactions on VT* 59, 9 (2010), 4268–4279.

[49] Nowakowski et al. 2015. Cooperative Adaptive Cruise Control (CACC) for Truck Platooning: Operational Concept Alternatives. (Mar. 2015).

[50] Open Source Vehicular Network Simulation Framework. 2019. veins.car2x.org.

[51] P. Papadimitratos et al. 2006. Securing Vehicular Communications - Assumptions, Requirements, and Principles. In *ESCAR*. Berlin, Germany, 5–14.

[52] P. Papadimitratos et al. 2008. Secure Neighborhood Discovery: A Fundamental Element for Mobile Ad Hoc Networking. *IEEE Comm. Mag.* 46, 2 (Feb. 2008).

[53] P. Papadimitratos et al. 2008. Secure Vehicular Communication Systems: Design and Architecture. *IEEE Comm. Mag.* 46, 11 (Nov. 2008), 100–109.

[54] P. Papadimitratos and A. Jovanovic. 2008. GNSS-based Positioning: Attacks and Countermeasures. In *IEEE MILCOM*. San Diego, CA, USA.

[55] P. Papadimitratos and Aleksandar Jovanovic. 2008. Protection and Fundamental Vulnerability of GNSS. In *IEEE IWSSC*. Toulouse, France, 167–171.

[56] Ploeg, Jeroen and Scheepers, Bart T. M. and van Nunen, Ellen and van de Wouw, Nathan and Nijmeijer, Henk. 2011. Design and experimental evaluation of cooperative adaptive cruise control. In *2011 14th International IEEE Conference on Intelligent Transportation Systems (ITSC)*. 260–265.

[57] Rabiner, L.R. 1989. A tutorial on hidden Markov models and selected applications in speech recognition. *Proc. IEEE* 77, 2 (Feb. 1989), 257–286.

[58] Rajamani, Rajesh. 2011. *Vehicle dynamics and control.* Springer Science & Business Media.

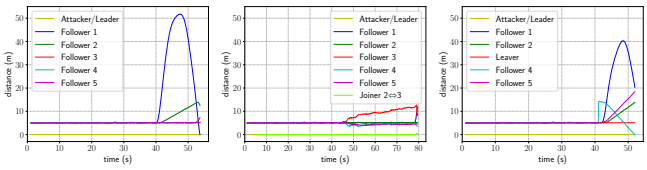
[59] Conrad Sanderson and Ryan Curtin. 2016. Armadillo: a template-based C++ library for linear algebra. *Journal of Open Source Software* 1, 2 (Jun. 2016), 26.

[60] Conrad Sanderson and Ryan Curtin. 2018. A User-friendly Hybrid Sparse Matrix Class in C++. In *ICMS*. South Bend, IN, USA.

[61] Santini, Stefania and Salvi, Alessandro and Valente, Antonio Saverio and Pescapé, Antonio and Segata, Michele and Lo Cigno, Renato. 2017. A Consensus-Based Approach for Platooning with Intervehicular Communications and Its Validation in Realistic Scenarios. *IEEE Transactions on Vehicular Technology* 66, 3 (Jun. 2017), 1985–1999.

[62] Gideon Schwarz. 1978. Estimating the dimension of a model. *Annals of statistics* 6, 2 (Mar. 1978), 461–464.

[63] Michele Segata. 2017. Platooning in SUMO: an open source implementation. In *SUMO User Conference*. 51–62.
[64] Steven So, Jonathan Petit, and David Starobinski. 2019. Physical Layer Plausibility Checks for Misbehavior Detection in V2X Networks. In *Proceedings of the 12th Conference on Security and Privacy in Wireless and Mobile Networks*. Miami, Florida, USA, 84–93.
[65] Hagen Stübing, Jonas Firl, and Sorin A Huss. 2011. A Two-Stage Verification Process for Car-to-X Mobility Data based on Path Prediction and Probabilistic Maneuver Recognition. In *IEEE Vehicular Networking Conference (VNC)*. Amsterdam, Netherlands, 17–24.
[66] Sun, Mingshun and Li, Ming and Gerdes, Ryan. 2017. A data trust framework for VANETs enabling false data detection and secure vehicle tracking. In *2017 IEEE Conference on Communications and Network Security (CNS)*. 1–9.
[67] D. Swaroop et al. 1999. Constant Spacing Strategies for Platooning in Automated Highway Systems. *J. Dyn. Sys., Meas., Control.* 121, 3 (Sep. 1999), 462–470.
[68] Seyhan Ucar et al. 2018. IEEE 802.11 p and Visible Light Hybrid Communication Based Secure Autonomous Platoon. *IEEE TVT* 67, 9 (May 2018), 8667–8681.
[69] R. van der Heijden et al. 2017. Analyzing Attacks on Cooperative Adaptive Cruise Control (CACC). In *IEEE VNC*. Torino, Italy, 45–52.
[70] R. W. van der Heijden, S. Dietzel, T. Leinmüller, and F. Kargl. 2018. Survey on misbehavior detection in cooperative intelligent transportation systems. *IEEE Communications Surveys & Tutorials* 21, 1 (Oct. 2018), 779–811.
[71] van Willigen et al. 2013. A Multi-Objective Approach to Evolving Platooning Strategies in Intelligent Transportation Systems. In *ACM GECCO*. 1397–1404.
[72] Pravin Varaiya. 1993. Smart Cars on Smart Roads: Problems of Control. *IEEE Trans. Automat. Control* 38, 2 (Feb. 1993), 195–207.
[73] W. Whyte et al. 2013. A Security Credential Management System for V2V Communications. In *IEEE VNC*. Boston, MA, 1–8.
[74] Michael Wolf et al. 2020. Securing CACC: Strategies for Mitigating Data Injection Attacks. In *IEEE VNC*. NY, USA.
[75] K. C. Zeng et al. 2018. All Your GPS Are Belong to Us: Towards Stealthy Manipulation of Road Navigation Systems. In *USENIX*. Baltimore, MD, USA, 1527–1544.
[76] Y. Zheng et al. 2015. Stability and scalability of homogeneous vehicular platoon: study on influence of information flow topologies. *IEEE TITS* 17, 1 (Mar. 2015).
[77] Zijia Zhong and Joyoung Lee. 2019. The effectiveness of managed lane strategies for the near-term deployment of cooperative adaptive cruise control. *Transportation Research Part A: Policy and Practice* 129 (2019), 257–270.
[78] YJ Zhou, HB Zhu, MM Guo, and JL Zhou. 2020. Impact of CACC vehicles' cooperative driving strategy on mixed four-lane highway traffic flow. *Physica A: Statistical Mechanics and its Applications* 540 (Feb. 2020), 122721.



(a) No Maneuver **(b) Join Process** **(c) Exit Process**

Figure 16: Leader Combined Speed (100 km/h) on Flatbed.

A COMPARING VEHICLE COLLISIONS

Fig. 16 shows the results of a combined speed falsification attack, performed by a malicious leader, with Flatbed during a static platoon, a join and an exit process. The attack without a maneuver causes the follower of the leader (i.e., the second vehicle in the platoon) to crash into the attacking vehicle. In contrast, during the join process, the joiner crashes into its predecessor upon entering the formation. During the exit process, the attack causes a vehicle downstream to crash into its predecessor.

B SECURE CACC MANEUVER PROTOCOLS

Join Procedure: A vehicle joining the platoon, a *joiner*, sends a request to the platoon leader; providing the platoon identity ($Id_{platoon}$), its desired position (Pos), the vehicle length (S), lane number (L), a nonce, the current timestamp, and its currently valid pseudonym to facilitate verification by the receiver. The joiner signs this message with the private key corresponding to its current pseudonym (step 17.1). Upon receiving the request and verifying the signature (and the pseudonym, if not done so before) (step 17.2), the leader checks the availability of the requested position (step 17.3). It then deactivates the lane change maneuver and crafts a new formation which includes the joiner. Then, the leader notifies the latter to approach the platoon, by providing the current speed (V), lane (L), vehicle length (S), type of controller (C), the platoon formation (F), and the approved position (Pos). The leader message is signed and sent to the joiner (step 17.4). Upon receipt of the message, the joiner verifies the signature, approaches the position, and notifies the leader (steps 17.5–17.7).

The leader verifies the *confirmation* (step 17.8) and proceeds to the next step depending on the type of joining process. In case of a tail join, it notifies the joiner to join from the tail (step 17.9). In case of a middle join, it notifies the platoon members about the maneuver (steps 17.10 and 17.13). The *jfollower*, the vehicle residing at the entry position, creates the gap for the joiner to enter. The vehicles following the *jfollower* anticipate the speed reduction and maintain the appropriate intra-platoon distances. Once the gap is established, the leader sends an authenticated notification to the joiner to enter the formation (steps 17.16–17.20). Once the joiner successfully enters the platoon, the leader notifies the platoon members about the new formation (steps 17.21–17.26).

During the middle-join process, the joiner needs to maintain its relative position (both to the jfollower and its 'predecessor') until the maneuver finishes (Protocol 1). To achieve that, the joiner calculates the distance based on its current and its predecessor's position (step 1.3). and the required distance based on the type of the controller (step 1.4). If the calculated distance is greater than twice the estimated gap (including its car length), or it is less than a predefined (parameterized) threshold, it aborts the joining process (steps 1.5–1.7). This distance estimation is executed until the leader notifies the joiner to change its lane and merge into the platoon.

Exit Procedure: As per Fig. 18, a vehicle that is to leave the platoon, the *leaver*, submits a request to the leader, digitally signed with the private key corresponding to its currently valid pseudonym (step 18.1). Upon verifying the signature, the leader deactivates any lane changes, checks if the exit is allowed, and updates the platoon formation (steps 18.2–18.3). It then notifies the leaver to exit the platoon (step 18.4). The leaver verifies the signature (step 18.5) and initiates the leave formation process (step 18.6). It then sends a message to inform the leader that it exited the platoon (step 18.7). The leader receives the confirmation, it validates the signature, and acknowledges the leaver (steps 18.7–18.8). The leaver changes the controller, e.g., switching to ACC and finally, it sends an acknowledgment to the leader (steps 18.10–18.11). Upon receiving the message, the leader verifies the signature and notifies all the platoon members about the new formation (steps 18.12–18.13).

Mitigating Privilege Escalation: Certain protocol implementations [17, 29, 49] require the platoon to be temporarily split in order to perform the maneuver. This, however, can lead to a privilege escalation attack as a malicious platoon member can trivially become a leader, e.g., by colluding with another vehicle who wishes to enter just ahead or with its predecessor who elects to leave the

Table 6: Notation Used in the Protocols.

AbortManeuverProcess()	Aborts the current maneuver
ApproachingPlatoon()	Approach platoon and maintain position
beacon	The beacon received through V2V
C	Controller
D	Distance
ExitPlatoon()	Remove internal platoon information
F	Platoon formation
γ	Maximum distance allowed before aborting
$\mathbb{G}$	The estimated intra-platoon gap
IncreasingSpace()	Increase distance based on controller
$Id_{req}, Id_{res}, Id_{platoon}$	Request/Response/Platoon Unique Identifiers
JoinProcessInit()	Performe pre-join steps
JoinFormation()	Change lane and update internal structures
K_v^i, k_v^i	Pseudonymous public/private key pairs
L	Lane number
LeaveProcessInit()	Deactivate maneuvers and validate exit
LeaveFormation()	Change Lane and controller
ManeuverUnderway()	Update internal structures and increase gaps
$(msg)_{\sigma_v}$	A signed message with the vehicle's private key
N, N'	Nonces
$(P_v^i)_{pca}, P_v^i$	A pseudonym signed by the PCA
Pos, V, S	Position, Velocity, Size
$ReceiveResFromLeader()$	Wait until the leader responds
$SetFrontVehicle()$	Update the local values of the predecessor
$Sign\ (Lk, msg)$	Sign a message with the private key (Lk)
$UpdateFormation()$	Change $vehicle_{ids}$, belonging to a platoon
Veh^i	Holds i's vehicle data, including Pos, V and Size
$Verify\ (LK, msg)$	Verify a message with the public key (LK)

formation. Our protocol, does not promote a member to temporarily lead and the platoon is never split during a maneuver. Instead, the vehicles downstream (from the join or leave position) either decrease their speed to create a gap during join (steps 17.9–17.20), or increase their speed to close the recently created gap after an exit (steps 18.5–18.13) while maintaining the platoon formation. This, essentially, eliminates the privilege escalation attack.

Protocol 1 DistanceEvaluator (by the Joiner)

```
1: procedure DistanceEvaluation(beacon, Veh, C)
2:     repeat
3:         D = beacon[Pos] − Veh_Pos − Veh_S^i
4:         𝔾 = GetPlatoonGaps(C, VehV)
5:         if D > ((2 × 𝔾) − γ) OR (D < γ) then
6:             AbortManeuverProcedure()
7:         end if
8:     until !ReceiveResFromLeader()
9: end procedure
```

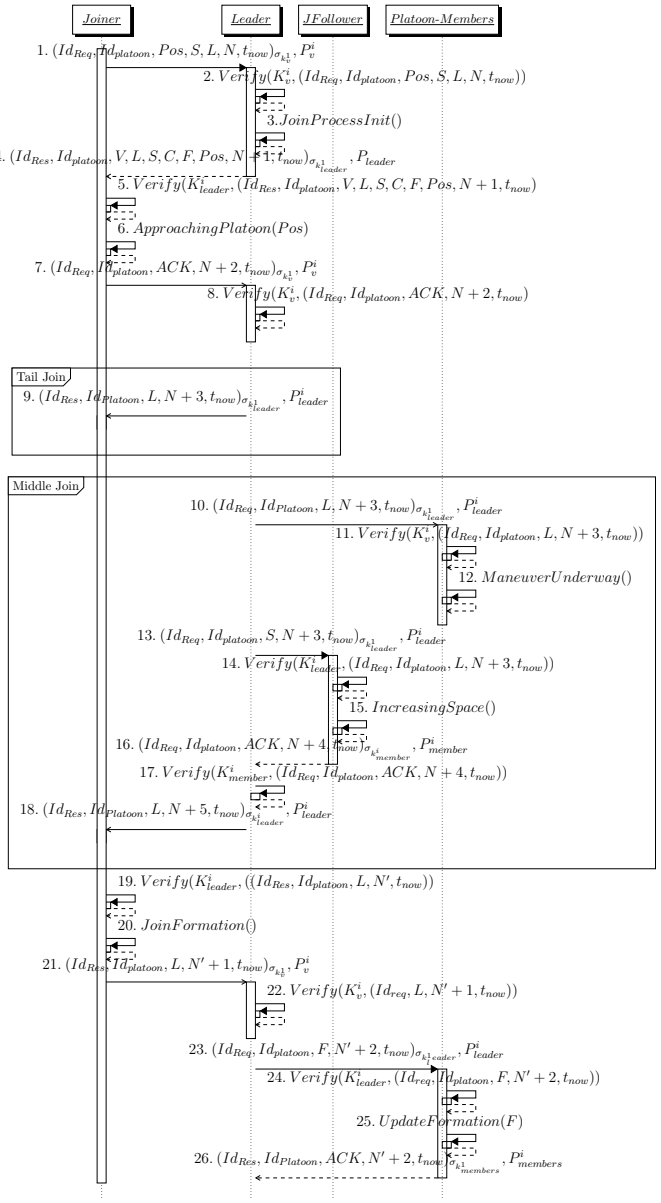


Figure 17: Vehicular Platooning: the Join Process.

C DETECTION VS MITIGATION

To avoid crashes (vehicle collisions), one can degrade from CACC to ACC (e.g., [31, 35, 69, 74]). Fig. 19.a shows that leveraging a kinematic fusion with a Kalman Filter to detect a gradual acceleration falsification attack (during *no* maneuvering) and employing such a countermeasure would prevent an accident (for PATH controller). But, such a countermeasure cannot mitigate colluding adversaries (Fig. 19.b): even though the attack is detected, simply degrading to ACC is not adequate to prevent a collision. This is caused by the processing delay of the MDS (1.5-3.9 sec.) [35]. A detailed investigation into the causes is part of our future work.

D CONTROLLERS COMPARISON

Table 7 compares controllers under different attacks in terms of the percentages of the total number of experiments that did not result into instability or collision. The higher the percentage is, the higher the resilience of the controller under attack. Ploeg is affected only by acceleration attacks, while Consensus and Flatbed are affected by position and speed attacks. However, the more sophisticated an attack is, the less resilient the controllers are.

Table 8 compares the instability of the controllers for different attack scenarios. Based on the experiments, PATH and Ploeg are unstable even under simple falsification attacks. PATH requires

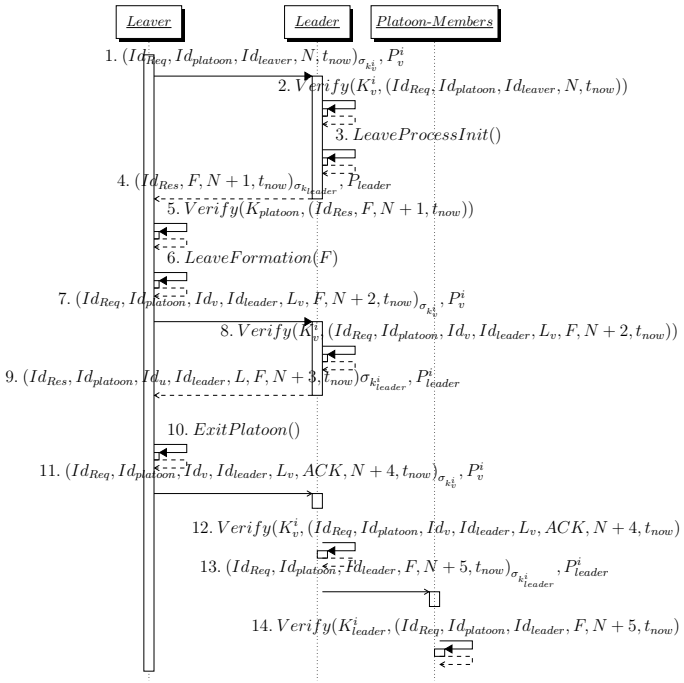
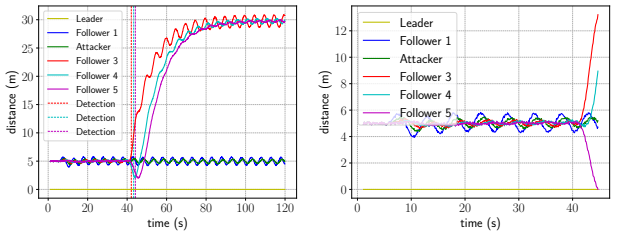


Figure 18: Vehicular Platooning: the Leave Process.



(a) Gradual Acc. Falsification **(b) Gradual Acc. Falsification With Colluder**

Figure 19: Kalman Filter Detection and ACC Mitigation.

speed and acceleration information disseminated by the predecessor; thus, it is susceptible to attacks that falsify these values. Ploeg, however, relies only on the acceleration information from the predecessor. As a result, it is not affected by any other falsified value, i.e., position and speed. Similarly, Flatbed is more prone to the speed falsification attacks by the leader. It is not, however, affected by speed falsification attacks by the predecessor. Consensus appears unstable, but it is the most crash-resilient controller (overall, see Table 9) because it utilizes information from all the vehicles in the platoon.

Table 9 shows the probability of a collision under different attack scenarios. PATH and Ploeg mostly result in a crash; Flatbed is more resilient, especially with the more complex attacks, but is still more crash-prone than Consensus, which outperforms all controllers. Simple and gradual position falsification attacks have the least impact as all controllers are resilient to them. Conversely, the combined position falsification attacks have the highest impact.

Table 7: Controllers Resilience Comparison (%).

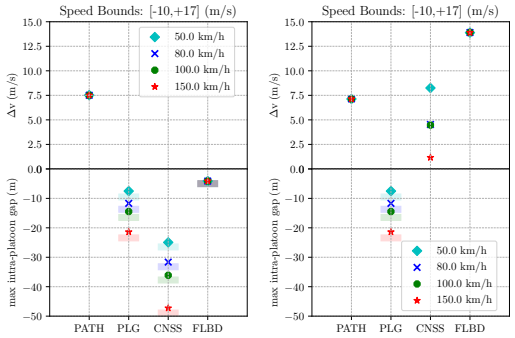
Controller	Simple			Gradual			Combined		
	P	S	A	P	S	A	P	S	A
PATH	100	0	20	100	0	0	0	0	0
Ploeg	100	100	20	100	100	0	0	0	0
Consensus	0	50	100	0	33.3	100	0	45.8	62.5
Flatbed	91.7	14.7	51.7	87.5	0	33.3	0	29.1	29.1

Table 8: Controllers Instability Comparison (%).

Controller	Simple			Gradual			Combined		
	P	S	A	P	S	A	P	S	A
PATH	0	2.9	17.5	0	0	0	0	0	0
Ploeg	0	0	20	0	0	0	0	0	0
Consensus	0	35.3	0	37.5	16.7	0	50	25	37.5
Flatbed	8.3	54.9	40	12.5	50	33.3	0	16.7	4.2

Table 9: Controllers Crash Comparison (%).

Controller	Simple			Gradual			Combined		
	P	S	A	P	S	A	P	S	A
PATH	0	97.1	62.5	0	100	100	100	100	100
Ploeg	0	0	60	0	0	100	100	100	100
Consensus	100	14.7	0	62.5	50	0	50	29.2	0
Flatbed	0	30.4	8.4	0	50	33.4	100	54.2	66.7



(a) Malicious Follower **(b) Malicious Leader**

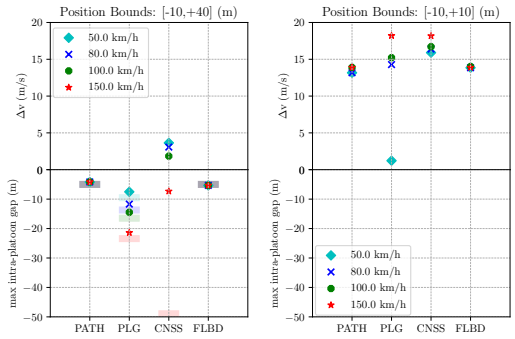
Figure 20: Attack Impact: Gradual Speed Falsification.

E INTELLIGENT ATTACKS DURING STATIC PLATOONING

Fig. 20 compares the gradual speed falsification attacks for different controllers. Fig. 20.a shows the effect when the attack is performed by a follower: only the PATH controller is affected and the victim crashes into the attacking vehicle in less than 6 seconds with a ΔV of 7.5 m/s (or 27 km/h). Fig. 20.b illustrates a gradual speed falsification attack by the leader: all controllers but Ploeg are affected. Among the affected ones, Flatbed has the highest crash severity

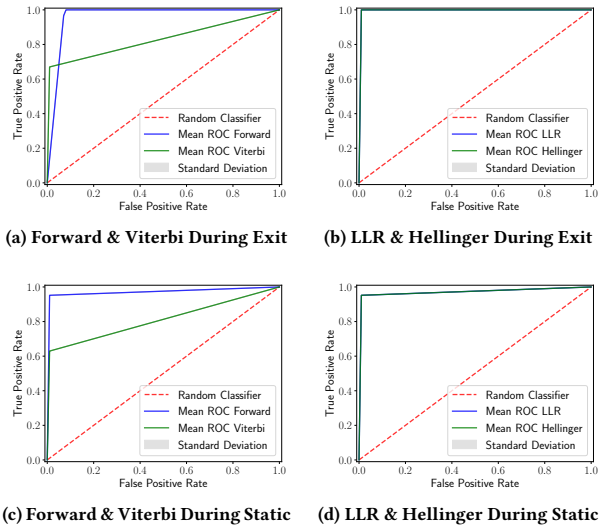
(≈50 km/h). For the Consensus controller, the higher the speed, the less severe the collision impact; this is due to the increased intra-platoon gaps, which allow Consensus to compensate for the increasing acceleration.

Fig. 21.a shows the effect of gradual falsification attacks by the leader: only the Consensus controller crashes when the speed is less than 100 km/h. Fig. 21.b demonstrates the effectiveness of a combined position falsification attack (by the leader). All controllers are highly affected, with collision severity ranging from 50 km/h to 65 km/h (13.8 m/s and 18 m/s in the figure). The Ploeg-controlled platoon, at the lowest speed, is the only discrepancy in terms of impact because the collision does not involve the immediate victim but rather the vehicles downstream.



(a) Gradual Falsification (b) Combined Falsification

Figure 21: Attack Impact: Leader Position Falsification.



(a) Forward & Viterbi During Exit (b) LLR & Hellinger During Exit

(c) Forward & Viterbi During Static (d) LLR & Hellinger During Static

Figure 22: ROC: Combined Position Leader Attack on PATH.

F RECEIVER OPERATING CHARACTERISTIC

Fig. 22 compares the Viterbi, Forward, LLR and Hellinger approaches, used by the anomaly detection component of the MDS, against a malicious leader. We choose to show the ROCs for the combined position falsification attack because it results in the biggest number of crashes among all controllers (See Table 9). LLR and Hellinger outperform both Forward and Viterbi; they achieve a TP rate close to 1 at a FP rate of 0.01. Forward achieves higher TP rate than Viterbi; albeit, slower during an exit maneuver. Forward suffers from low precision (higher FP) rates affecting its score; increasing the detection threshold ultimately reduces its overall (F_1) detection capabilities. It should be noted here, that the standard deviation in all cases is too small to be observable; all four methods achieve almost the same rates every time they are executed.

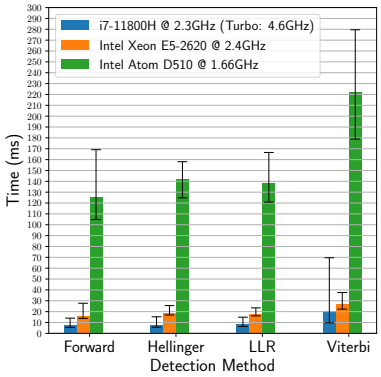


Figure 23: Processing overhead: Comparison of CPU timings

G MEASURING PROCESSING OVERHEAD

In Fig. 23 we present the processing time required by each MDS method, when run in three different machines (with Ubuntu 20.04). The output is a detection of attack based on an observation (sliding) window. Each bar quantifies the average values recorded; the lower and upper margins quantify the best and worst value respectively. The 2.3 and 2.4 Ghz Central Processing Units (CPUs) correspond to regular workstations and the third CPU, at 1.66 Ghz, corresponds to a modest On-Board Unit (OBU) [1]. Even though the OBU is several times slower than the more powerful units, in the worst case scenario, even with the slowest MDS method, the time needed for a decision is 280 ms. In the case of LLR, only 130 ms are required. Considering the dissemination frequency of CAM messages (100 ms) and the sliding detection window of 1 second, this time is not prohibiting. Further, this allows for even smaller windows depending on the safety application. Moreover, any in-car architecture that performs CPU intensive operations in another on-board machine other than the OBU, handling V2X, can handle MDS with lower latencies and operate with broader windows. Such considerations are part of future work.