Investigating Attacks on Vehicular Platooning and Cooperative Adaptive Cruise Control

KONSTANTINOS KALOGIANNIS

Master's Programme, Computer Science, 120 credits
Date: February 12, 2021
Supervisor: Mohammad Khodaei
Examiner: Panos Papadimitratos
School of Electrical Engineering and Computer Science
Swedish title: Undersökning av attacker på fordonståg och kollaborativ adaptiv farthållning

Investigating Attacks on Vehicular Platooning and Cooperative Adaptive Cruise Control / Undersökning av attacker på fordonståg och kollaborativ adaptiv farthållning

Abstract

Autonomous vehicles are a rising technology that aims to change the way people think about mobility in the future. A crucial step towards that goal is the assurance that malicious actors cannot instigate accidents that could lead to damages or loss of life. Currently, vehicle platoons, that is vehicles cooperating together to increase fuel saving and driver comfort, are used in limited environments and are the focus of research aimed to make them suitable for real-world wide usage. In that regard, guaranteeing that the vehicle is able to operate alongside other entities, autonomous or not, in the traditional sense is not adequate. The computer systems involved can be the target or the source of a malicious act without the knowledge of the operator in either case. In the context of platooning, these acts can have devastating effects and can originate either from other vehicles on the road or from within, from compromised vehicles that are part of the formation.

In this thesis, the focus is centered around the latter. We investigate jamming and data falsification attacks that aim to either destabilize the platoon, thus, reducing its benefits or provoke an accident. These attacks are more difficult to discern and will range from simple falsification attacks to more complex ones that aim to bypass defensive mechanisms. In that sense, we direct our experiments against the platoon maneuvers that are a core functionality of platooning and are required for its nominal operation. The results of this analysis show that several attacks can lead to accidents with position falsification being the most productive. It is also demonstrated that a malicious leader can pose a serious threat to the viability of the platoon because of his unique capability of interacting with all the platoon members. Attacks during the platoon maneuvers are demonstrated to pose a threat, not only to the stability of the formation but also the nature of the platooning application itself. This is achieved by effectively isolating the platoon from potential joiners.

Keywords

V2V security, Injection Attacks, CACC, Anomaly Detection

Sammanfattning

Självkörande fordon är en framväxande teknologi med mål att ändra människors framtida inställning till mobilitet. Ett kritiskt steg mot målet är att försäkra sig om att aktörer med ont uppsåt inte kan orsaka olyckor som kan leda till skador eller dödsfall. För närvarande används fordonståg, alltså fordon som samarbetar för att minska bränsleförbrukning och öka körkomfort, i avgränsade miljöer med fokus på att anpassa dessa för verklig användning. Att garantera att fordonet kan köras tillsammans med andra enheter är då inte tillräckligt eftersom dessa system kan bli mål för externa och interna attacker som kan ha förödande konsekvenser.

Denna uppsats fokuserar på det senare fallet och undersöker interna datafalsifierings- och frekvensstörningsattacker avsedda att destabilisera fordonståg i syfte att minska deras fördelar eller provocera fram en olycka. Dessa attacker är svåra att urskilja och inkluderar allt från enkla falsifikations-attacker till komplexa attacker som syftar till att kringgå specifika försvars-mekanismer. Med det i åtanke inriktar vi våra experiment mot de manövrar som är en del av fordonstågens grundfunktionalitet och krävs för deras nominella drift. Resultaten av arbetet visar att under fordonstågmanövrar så kan flertalet av de utvärderade attackerna orsaka olyckor och att attacker genom förfalskning av position var speciellt förödande. Vi har även påvisat att en fordonstågsledare med ont uppsåt utgör ett speciellt allvarligt hot mot fordonstågets funktionalitet på grund av dennes unika möjlighet att interagera med alla medlemmar. Attacker under manövrar har visats utgöra ett hot, inte bara mot stabiliteten av formationen, men även mot de grundläggande egenskaperna hos systemet själv såsom att isolera fordonståget från nya medlemmar.

Nyckelord

V2V säkerhet, Falsifieringsattacker, CACC, Avvikelse Upptäckt

Acknowledgments

Another chapter of my life comes to an end and that would not be possible if not for the support and guidance from the people near me; the patience they have shown is much appreciated. The first person to acknowledge especially in the context of completing this work is my supervisor, *Mohammad Khodaei*, who through countless meetings has helped me focus my interest, posed questions that led me to learn new things and guided me in exploring new avenues not only for the this work but also for the field of network security in general. A special thanks should be given to *Prof. Panos Papadimitratos*, leading the Networked Systems Security (NSS) group at KTH, who gave me the opportunity to delve into the subject at hand and brought this thesis into fruition with his crucial insights.

That said, a simple thank you would be an understatement when it comes to my beloved family and friends who were with me in this journey and helped me carry on despite all the unforeseen difficulties of this year. As a final note, the organized and welcoming environment of KTH and Swedish culture was a beneficial factor throughout the two years of my stay in Stockholm.

Stockholm, February 2021
Konstantinos Kalogiannis

Contents

List of Figures

List of Tables

List of Source Code

Chapter 1

Introduction

1.1 Background

With the advent of autonomous systems in the automotive industry, platooning is regarded as a promising concept to make road transportation of goods and individuals safer and more comfortable for all the parties involved. It is also considered an encouraging solution in reducing the fuel consumption [1], bringing down both the cost for consumers and the vehicle operators. The basic premise of platooning relies on the fact that vehicles cooperating on the road can achieve higher speeds while simultaneously having smaller gaps between them. This results in raising the throughput of the road they ride, originally proposed by Pravin et al. [2]. This cooperation is an improvement to the Adaptive Cruise Control (ACC) functionality of vehicles, called Cooperative Adaptive Cruise Control (CACC), and is achieved through message passing; their enclosed data are consumed by the different CACC controllers operating in each car, creating a platoon formation. These convoys can operate in a reliable way as a homogeneous system with the first car functioning as the platoon leader and the trailing vehicles following without the need for human interaction, thus freeing the subsequent drivers [3] [4]. Typically the vehicles operate in a column, although that is not the only possible formation as seen in other fields such as agriculture or the military [5]. Despite the usefulness of this approach, it is apparent that its safety is of paramount importance if we consider that humans are involved in the scheme, even though they may not directly influence the functionality of the cars. In this case, the benefit of platooning is also one of its most glaring problems, shorter gaps between the vehicles reduce the response time, especially when the driver is not actively driving the vehicle.

To make platooning possible though, a whole ecosystem, called Intelligent Transportation System (ITS), exists with autonomous vehicles playing only a small part in it, as described in [6] by the European Telecommunications Standards Institute (ETSI). The whole infrastructure is composed from systems like Road Side Units (RSUs), ITS service centers, which are responsible for handling the different RSUs, and vehicles. The application layer model of Vehicle to Everything (V2X) is comprised of several types of communications. In cases that it only involve cars, it is called Vehicle to Vehicle (V2V), while in cases where the car needs to get in touch with a RSU it is named Vehicle to Inrastructure (V2I); when the opposite is true, it is called Infrastructure to Vehicle (I2V). These systems aim to increase the traffic flow, reduce accidents and alert vehicles nearby about emergency situations. For example, when an accident occurs on the road ahead, the traffic will come to a stop, but emergency services alerted through the ITS ecosystem, still need an available route to reach the scene. Through V2X messages, or beacons, this information can be passed from cars to RSUs and back to vehicles to divert them to alternate routes and decongest the situation. The above is possible thanks to well defined standards which make their adoption easier by the different automotive companies that operate not only in Europe, but the world in general. In that regard, the standard does not strictly separate or merge, CACC from platooning, but in our work we use the terms interchangeably. Relevant research has shown that vehicles operating together using CACC can have comparable cost benefits [7], thus, supporting the cases of controllers that may not be regarded as platooning.

The basis of Vehicular Communication (VC) is a variant of the IEEE 802.11p standard and takes place in allocated channels in the band of 5.9 GHz, which is dedicated only to its use [8]. The network packets, or beacons, use the data link layer for their transmission and are usually handled as broadcasts at the medium access control layer [9]. The various ITS systems transmit two types of beacons, called Cooperative Awareness Message (CAM) and Decentralized Environmental Notification Message (DENM) [8]. The former is transmitted periodically between the vehicles but does not always carry the same information. Data that can change quickly are always transmitted and thus are disseminated with a 10 Hz frequency, that is every 100 ms, and include information such as speed, acceleration and direction; while less dynamic data, like vehicle roles or path history, can be sent less frequently, every 500 ms [8]. DENMs on the other hand, are event-triggered messages broadcast in a multi-hop way and contain information relevant to the local environment, like road hazards or traffic jams. For the purpose of securing the VC, the

standards bodies ETSI [6] and IEEE 1609.2 WG [10] have chosen Public Key Infrastructure (PKI) which in the context of ITS is called Vehicular Private Key Infrastructure (VPKI) [11] [12] [13] [14] [15]. The above adoption is also helped by the harmonization carried on by Car 2 Car Communication Consortium (C2C-CC) [16], a joint-force of vehicle manufacturers, research institutions and engineering companies.

Platooning achieves its goals by utilizing several of the subsystems of ITS and its success relies on certain components behaving in the appropriate manner. We can categorize them in the following way:

- Message passing between all the vehicles and/or the RSUs, in short, V2X.
- Vehicle sensors responsible for determining the current location and the range between the vehicles.
- Fallback mechanisms to handle faults.

All three of them, can either malfunction or be the target of adversaries who want to subvert them for their advantage. V2X is the first on that list and platoon management protocols require it to behave as expected in order to keep the vehicles in a cohesive formation. Thus, attacks that target it, like Denial of Service or Wormhole [17] attacks are detrimental to its functionality and require countermeasures to prevent them. In that regard, several security requirements exist in the form of message authentication and integrity [18] [19]. On the other hand, the car sensors are responsible for dictating its independent course of action even when there is a network failure, but we will not delve into their hardware proofness in this work. Finally, in accordance with the guidelines of ISO 26262 [20], safety mechanisms exist to guarantee the correct and reliable operation of the vehicle even in environments were all the other elements used in CACC are malfunctioning. This can range from increasing the inter-platoon gap, in order to accommodate the reduced autonomous functionality, to giving full control to the driver while making sure that his reaction time is taken into consideration.

Despite these safety measures, autonomous functionality does not exist in a vacuum. It cannot be assumed that the cars operate without any interference to their network or their homogeneity (by interleaving non-automated vehicles) on the road [21]; platoon management protocols do not take that into consideration, they typically assume a special lane dedicated only for CACC operations [22]. At this point, we should define the three basic properties that are crucial for a nominal platoon operation, which depending

on their combination, can either hinder or promote the platoon's functionality. Figure 1.1 provides a good depiction of this, with two platoons of size three operating one after the other and their members interacting through V2V communication.

- Intra-platoon gap: This gap refers to the separation that exists between each car. It affects, on one hand, the stability of the platoon, that is the propagation that disruptions have from start to finish, while on the other hand it increases the road throughput.
- Platoon size: The size of the platoon can increase the utilization of the road, but inadvertently, is correlated with propagating platoon instability from one platoon to the other.
- Inter-platoon gap: This is usually a far bigger gap than the first and the reason is twofold. Firstly, it guarantees that there is enough space to perform all the maneuvers, especially the ones increasing the size of the platoon, and secondly, that the platoons can not collide.

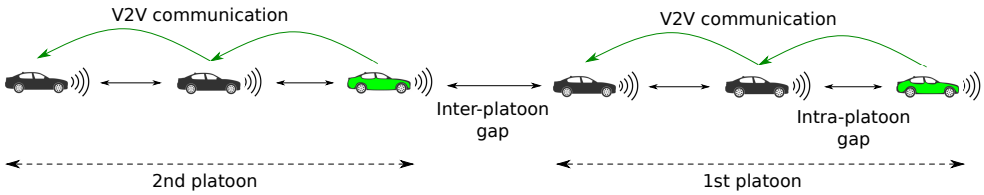


Figure 1.1: Two platoons of size three along with the necessary gaps required by the platooning application.

1.2 Research Question

The research question can be formulated as the systematic investigation of attacks on vehicular platooning with the purpose of degrading the platoon stability and decreasing its efficiency. In this work, we conduct jamming and falsification attacks on four different CACC controllers, i.e., PATH, Ploeg, Consensus and Flatbed (described in Section 2.3.3), and measure their susceptibility. All of them have a distinct mode of operations, giving us the opportunity to find small details that an intelligent attacker can use to maximize the effect of his actions. An internal adversary can jam the communication channel to disrupt the propagation of the beacons, in critical

moments, in order to cause an accident. Moreover, he can also falsify the transmitted data to induce instability into the platoon. In our work, we consider the dissemination of falsified speed, acceleration and position by a malicious actor residing, either inside the platoon or being its leader. To make our experiments more thorough and devastating for the platoon, we try to envision falsification attacks that an intelligent actor would perform. In that regard, we devised several of them by combining all the kinematic properties of the vehicles, which can be difficult to detect by defensive mechanisms that rely on data plausibility and consistency. Additionally, platoon applications require the existence of maneuvers, like join and exit, to dynamically change their formation. These processes are also susceptible to falsification attacks and thus we regard them as valid targets, which has not been investigated in the literature. Consequently, our question devolved into the following:

- What are the effects of injection attacks on the performance of the platoon controllers when the attacks are perpetrated during a maneuver?
- How is the platoon stability affected by network attacks regardless of the position of the attacker?
- What is the severity of the collisions and how they can affect the injury potential of the passengers.
- Can misbehavior detection function in the dynamic environment of platoon formations?

1.3 Contributions and Thesis Objective

The main purpose of this work is to provide a deeper understanding on the effects that internal attacks have on platoon formations. Considering the potential for material damages and human loss of life, it is imperative that these autonomous systems work nominally under all conditions. To achieve this, apart from developing our experiments, we also made several contributions that we outline below which can be valuable assets for any research done in this area.

Our work provides a unified approach, in contrast to previous work in the area, in conducting experiments that target the convoy without any limitation, either to the attacker's position, his chosen type of falsification attack or the existence of collaborators. The malicious actor can be the crucial leader or an incoming vehicle entering the platoon. He can act intelligently or

not, depending on the circumstances, or he can take action together with a collaborator maximizing the platoon instability and the potential for injuries. We demonstrate, that attacks performed in conjunction with a maneuver are possible and can be devastating despite the existence of countermeasures, proving that the dynamic nature of these convoys can pose difficulties in fine-tuning mechanisms that seem to work in static environments. To make the above come to fruition, we also contributed by expanding the software tools used in this work. We extend the already present join maneuver with the ability to include any joining position, which has not been shown in the literature despite its potential [23, 24, 25] and function under any controller; previously it supported only PATH. Our contributions also include several alterations in the maneuver's code-base, making it more robust and complete for further extensions. The scenarios were also extended with the implementation of the exit maneuver which we used in our tests. Finally, we adapted the kalman filter defensive mechanism, described in Section 2.4.3, and evaluated it with our experiments to find out its constraints.

These contributions aim to assist the research community by expanding its knowledge in the space of vehicular network attacks, and any vehicle manufacturer in taking the correct course of action when designing their vehicle systems and their control algorithms. In our case, the goal is to produce the following deliverables that promote our objective by showcasing the effect falsification attacks have on the platoon's operation; including measuring their severity in regards to their potential for personal injury.

- The kinematic properties of the vehicles, like speed and acceleration, provide a base understanding of how each controller responds to the various falsification attacks.
- The intra-platoon gaps reveal tangible results from the attacks, proving either platoon instability or inevitable crashes.
- The abrupt change of speed, for a vehicle involved in a crash, indicates the severity the attacks can have on vehicle passengers [26].
- The number of crashes per controller per platoon's speed, give an extensive insight into the effectiveness of the attacks. In other words, we provide a qualitative assessment in regards to the susceptibility of the currently available controllers.
- A detailed hitmap from the perspective of the malicious actor which provides the baseline for further research in controller design and misbehavior detection.

1.4 Methodology

This project uses both a qualitative and a quantitative methodology. The latter is more prevalent as we try to compare the different CACC controllers and find their limits under different scenarios and attacks. The induced randomization in the input data (sensors) also supports this approach. In that regard, we followed the methodology used by previous research in the area we are delving into. The line blurs when we identify several use cases (from those experiments) and we try to selectively examine them and solve them. The analysis we perform on the data to produce the Hitmap table (Table 4.4), which also includes a very crucial assumption, is an example of the qualitative methodology we followed in certain cases. Moreover, we compare several aspects of our work with previous research done in this area and we expand those tests in order to get a better understanding and reach a new consensus in the space of falsification and jamming attacks.

1.5 Ethical Considerations

The nature of this work involves performing attacks against systems that are responsible, apart from fuel-saving and cost efficiency, for ensuring the safety of travelers on the road. Such knowledge can always be used in nefarious ways in order to subvert the system's nominal functionality which is something we are aware of. The defensive mechanism tested here provides a base remedy for the attacks discussed but it is not a panacea. Our work, ultimately, aims to increase the awareness for the problem and guide future research based on our contributions. Thus, in the future, the developed defensive mechanisms and CACC controllers can be adequately prepared to mitigate them.

1.6 Structure of the Thesis

The rest of the dissertation is structured in the following way. Chapter 2 presents all the relevant background needed to understand how platoons work, from their management system and controllers, to how the vehicles can discern their location on the road. Then, the chapter delves deeper into the related work, the attacks that have already been performed, their assumptions and their results. Chapter 3 discusses our system model and describes the tools and the methodology used in order to generate the results of this thesis. The implementations of the attacks and the platoon maneuvers

needed to perform the experiments, are also reported and we showcase various scripts implementations, used for manipulating the vast data generated by the simulations. The chapter finishes by detailing the methodology used to validate our results. Chapter 4 starts by exploring our educated hypotheses, based on previous work and the systems in use, and confirms or disproves them by illustrating the wide range of the different plots that are available. Finally, Chapter 5 provides a reflection to this chapter, along with a summary of the knowledge gained through creating and performing the attacks, and examining their products. As a final note, we argue for potential followup work that can be made in the area.

Chapter 2

Background

In this Chapter, a general background of the platooning ecosystem is given in order to familiarize the reader with its components and offer him a better understanding of the nature of problems that need to be solved. This also helps to direct him, eventually, at the particular problem this report wants to tackle. The chapter is structured in five sections outlining all the necessary information needed to understand this work.

The first section tries to familiarize the reader with the basics of VC security and how ITS preserve their privacy. After that, the focus shifts to one of their applications, specifically platooning, and how it works. Section 2.2 aims to achieve this by dividing the reading into a section describing how maneuvers work, in Section 2.2.1, and the operation of the platoon management protocols, in Section 2.2.2, in regards to VC. We also take a closer look on the internals of each car, in Section 2.2.3, discussing the necessary fault tolerance mechanisms needed. Section 2.2.4 outlines a study measuring the role that mixed traffic can have on road throughput and at which point it is prudent to separate autonomous traffic in a different lane. The next section, 2.2.5, gives some background information on how the vehicles can discern their location on the road depending on the architecture; this functionality is critical both in therms of security, by identifying fake nodes, and in terms of privacy, less honest parties can leak the user's location.

Section 2.3 focuses in describing the different controllers that are used in this work. We take into account four of them and we elaborate on how they achieve their goal depending on their information flow topologies and the intra-platoon policies they utilize.

Section 2.4 digs into the related work for this thesis and what has already be done by other researchers. The discussion follows a chronological order

that includes the necessary bits that lead to this project; their assumptions, their limits and finally their results.

This chapter concludes with Section 2.5 in which we try to give a very condensed report of the previous work along with their shortcomings that inspired ours.

2.1 Vehicular Communication Security and Privacy

The standardization bodies mentioned earlier, along with the ongoing research in the topic, try to safeguard the ITSs from external attackers and make the architecture work in a way that also preserves the privacy of each individual component. This section elaborates more on the security implementations of VC and the safety assurances that VPKI provides.

VPKI is distinguished from a traditional PKI in two main aspects; the volume of certificates in the system and the balance between privacy and efficiency [11]. Considering the number of vehicles on the road this is not an easy task to tackle and is the focus of continuous research. VPKI relies on a set of Certificate Authority (CA) that issue certificates for connected devices in order to prove their identity; at the root of this scheme lies the Root Certificate Authority (RCA) which is responsible for issuing the certificates used by all the devices and infrastructure in the scheme. Each car, at first, needs to register with a long term authority called Enrollment Certificate Authority (ECA) [11] or Long-Term Certificate Authority (LTCA) [13] and uses these credentials, or tickets, when it contacts the location-relevant Pseudonymous Certificate Authority (PCA) to request the generation of its pseudonyms. These pseudonyms are then used to authenticate the vehicle to others on the road without exposing its true identity. The list of authorities also includes a Resolution Authority (RA) which is responsible for processing requests that reveal the long-term identity of a vehicle [13]; such a mechanism is necessary in order to revoke certificates of misbehaving vehicles or vehicles that reached their end of life.

The pseudonyms that the vehicle receive from the PCA need to first be generated. There are two policies in that regard. The pseudonyms can either be generated at the LTCA, where their time to live is decided, or by the PCA [14]. When the later generates them, it uses the start and end times provided by the LTCA. This way, the LTCA does not know the pseudonyms that correspond to the real identity; the only knowledge it has is the time frame. At the same time,

the PCA can not link the old pseudonyms of a vehicle with new ones. The only way for a vehicle to get new pseudonyms is to acquire a token. The token is provided by the LTCA, thus no authority obtains all the information needed. All the pseudonyms are accompanied by a time-frame and are generally short-lived. Otherwise, anyone observing a specific vehicle on the road for a longer period of time would be able to create a pseudonymous profile. A profile that always starts in a specific location, for example, corresponds probably to the home address of an individual [27]. Even in cases where the pseudonyms change every tenth message, in the presence of a perfect attacker, one that can capture all the messages, the probability of linking the pseudonyms is very high. In [27] they achieve this by combining a kalman filter along with Multiple Hypothesis Tracking (MHT) to predict the next positions of the vehicles. The algorithm works by creating associations between the incoming data and uses the filter to predict the position and speed of the vehicle for each step. This way each data point is tracked with a value that corresponds to the probability of the predicted new state being the measured value.

Several proposals have been made to guarantee the unlinkability of the pseudonyms. Vehicles for example can follow a scheme where they only change their pseudonyms inside a period of "silence"; where they do not transmit any message, including CAMs. This period corresponds to slow traveling paces, e.g., under 30 km/h or intersections [28]. This approach however reduces the awareness of the vehicles, leading to an increased probability of accidents. Another solution is the use of mix zones where the vehicles can change their pseudonyms securely [29]. An external eavesdropper can only perceive the vehicles entering and then exiting the area with different pseudonyms. The resilience of this approach, though, relies on the driver population and a uniform entering and exiting from the area. A novel approach in regards to these zones is presented in [30] [31] where the nearby RSUs are creating fake CAM messages signed by pseudonyms issued to imaginary vehicles. This way the traffic inside the area is artificially increased making it harder to discern the real vehicles exiting the zone.

This whole infrastructure aims to limit the information anyone has of a particular vehicle but has also been proven to be ineffective when there is collaboration between the authorities [14]. Research has shown that even though these parties can be trusted, the lucrative information that they gather can be tempting and so they are defined as *honest-but-curious*, that is, they execute the protocol correctly but they also gather information for their own reasons [12]. To mitigate pseudonym linkage and also some forms of Sybil attacks [32], Khodaei et al. [13] [14] [33] propose an architecture of CA

that guarantees both without the need for extra intra-VPKI communication as in [34]. The latter is achieved through the LTCA, who maintains a record of the tickets issued for each car and thus guarantees the existence of a single valid ticket; previous solutions in this topic presumed the existence of secure hardware in each vehicle [35]. Their work is further developed in [15], where the system is leveraged to work in the manner of System-as-a-Service, called Vehicular Private Key Infrastructure as a Service (VPKIaaS). This distinction makes the infrastructure more scalable and resilient against benign failures and attacks that aim to deplete the available resources.

All these pseudonyms and certificates, though, create a new problem for Vehicular Ad Hoc Networks (VANETs), that is, the need to revoke them when a case arises; such a case can be a detection of a continuous attack inside the platoon. Typically, this is accomplished through the use of Certificate Revocation Lists (CRLs) that are disseminated to nearby vehicles in order to revoke previously accepted digital signatures. In one hand, the approach followed in [36] combines block ciphers and a structure called bloom filters* to achieve CRLs sizes that grow linearly; because of their probabilistic nature, bloom filters are susceptible to *chosen insertion* attacks that aim to increase the chances of false positives. On the other hand, in [38], the constructed scheme avoids this size overhead and guarantees privacy even in the presence of *honest-but-curious* certificate authorities. Their work is also taking into account that vehicles' resources such as computational power and network bandwidth are, first and foremost, needed for safety critical functionality. They achieve this by tackling the problem from a *vehicle-centric* scope; the CRLs needed by each car corresponds to those that match the time frame and the region of the trip, which does not need to be known by the certificate authority. In a followup work [39] they also consider the possibility of temporarily revoking pseudonyms belonging to misbehaving or malicious nodes until they are deemed benign, again, without diminishing the privacy of the rejoining party.

Several more aspects can be found in the scope of security, though, that surpasses identity and credentials management. For example, network layer attacks can involve Distributed Denial of Service (DDoS) or radio jamming that can be prevented by channel switching and frequency hopping. Some

* Bloom filters are a space-efficient structure described in [37] that holds a bitmap which is the result of successive hashes on a piece of data. All the relevant bitmaps, in our case the pseudonyms of each car, are then combined in an bit-wise OR operation and stored. This bitmap is then checked against the computed bitmap, generated with the pseudonym of the sender when a beacon is received, in order to validate if it is part of the structure, thus validating the message in a rapid manner.

other forms of DDoS involve the flooding of the network with fake beacons to overwhelm the On Board Units (OBUs) of the cars. Shared beacon, between neighbors, verification and periodically validating the beacons of cached vehicles provides a good remedy against such an attack [40]. Furthermore, the network is not the only medium that can be compromised; the vehicles themselves can also be affected. This can range from the sensors measuring the intra-platoon distances, to the vehicle software implementing some critical functionality [41]. The former poses a serious problem to tackle because *tamper-proof* hardware is not always available or it can be expensive to deploy. Certain mechanisms exist, in such cases, that can make the hardware *tamper-resistant* [41] [42]. In [43], a serious effort is made to provide a comprehensive list describing all the relevant attacks that can occur in ITS along with potential countermeasures and gives some useful insight in the field of automotive security.

2.2 Platooning

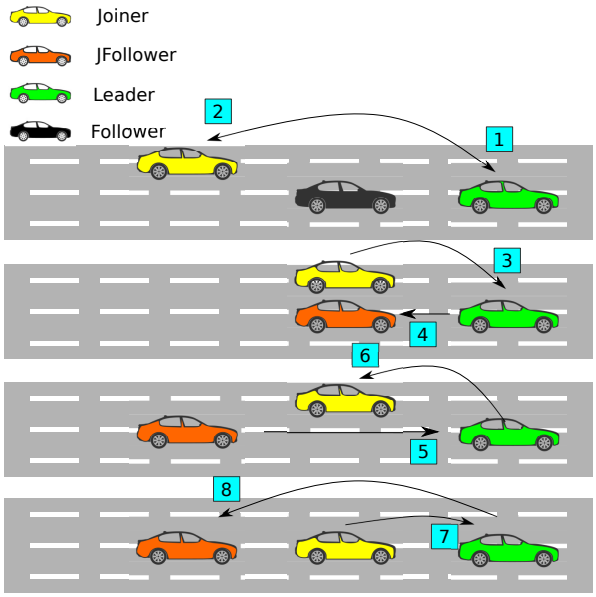


Figure 2.1: Join maneuver initiated by a passing vehicle aimed for the middle of the formation.

2.2.1 Maneuvers

Platoons form, operate and dissolve with the use of specific maneuvers. Regardless of where the platoon is formed, in a highway rest stop or the road, one of the vehicles is assigned to be the platoon leader. The Platoon leader is then responsible for dictating the CACC functionality everyone will follow and it is the target of platoon maneuver requests from vehicles who want to join or exit the convoy. He is also responsible for making sure that the properties of the platoon are kept and will deny any maneuver that tries to increase the platoon size from the predetermined value [44]. There are several maneuvers supported, but we discuss the *Join* and *Exit* as they are the building blocks for the others.

While traveling on the road, the leader of an already established platoon can elect to open the platoon to potential joiners. To do that, he periodically disseminates the platoon's status, namely that it accepts vehicles [45]. When a vehicle wants to enter such a platoon, it, the *Joiner*, first needs to query the leader if a *Join* is possible at the position he requests, denoted as #1 in Figure 2.1. This index value should not be taken lightly, since research has shown that vehicles closer to the leader enjoy improved fuel consumption and thus, the economic incentives is to be closer to him [46]. Additionally, middle joins have been shown to promote platoon stability [23] [24] due to the grouping of vehicles heading to the same, or near, destination and safety by placing less dynamic vehicles on the rear [25]. If the maneuver is approved, by the leader, the *Joiner* needs to approach the platoon by increasing its speed to catch up with it. If he chose a position in the middle of the platoon, he needs to align itself next to the vehicle which occupies his selected position; otherwise, he positions himself behind the platoon tail. At this point, the protocol diverges depending on the entry position.

- End of platoon: The *Joiner* notifies the leader that he is in position and requests for the platoon properties. The leader then communicates these values back to him and the latter changes its operational mode into the provided CACC one. To finish the protocol, the platoon leader notifies every vehicle in the platoon of the new formation.

- Middle of platoon: Because space needs to be created in the middle, certain operations need to happen first. When the *Joiner* reaches the correct position, he sends a message to the platoon leader to notify him. The leader, then, sends a message at the correct car in the platoon to make space; this car, called *JFollower* from this point forward, currently

occupies the requested position and needs to increase its intra-platoon gap. In order to preserve the CACC advantages, the *JFollower* only changes its operational values in regards to the intra-platoon distances in order to increase the gap. When this operation finishes, the *Joiner* can change his lane and enter the platoon before notifying the leader of this change. Finally, the platoon leader instructs all the cars in the platoon of the new formation; the *JFollower*, at this point, reverts its values to the nominal ones. An illustration of this is presented in Figure 2.1 where a car initiates a join in the middle of a platoon (containing two vehicles) while on the road.

Conceptually, a merge maneuver functions in the same way, with the only distinction being that the *Joiner* is also leading a platoon. In comparison, the exit maneuver is much simpler and is used as a building block for split, as discussed in [22]. A vehicle that wants to leave the platoon notifies the leader who then accepts or denies the maneuver. If the maneuver is accepted, the *Leaver* has to just change its lane and its operational mode and inform the responsible party (the leader) of the completion of the maneuver. The leader then notifies all the remaining vehicles about the event. At this point, the vehicle that resided behind the *Leaver* speeds up in order to conform with the spacing or headway imposed by its controller.

To bring the former point to fruition, we can investigate a maneuver that tries to split a platoon. When the platoon size exceeds a predetermined number, the leader will initiate a split maneuver involving the vehicles at the tail of the platoon. At first, he sends a message to the first in line of those vehicles to notify him of the decision to split the formation and be the new leader. Upon receiving the response, the leader notifies the vehicles behind the chosen new leader and tells them to change their structures to account for this event. At this point, an exit event takes place and the platoons split. More insights on the actual implementation of the *Join* and *Exit* maneuvers are given in Chapter 3 and in Appendix A.

2.2.2 Platoon Management

Typically, the leader is the one that initiates a message sequence in order to pass information and orders down the platoon hierarchy. In this scheme, messages flow in one direction, although, there is no standard approach and other strategies exist utilizing bidirectional communication and direct messages between vehicles [47], that is without traversing each car to reach the target. A comprehensive analysis of the different information flow topologies

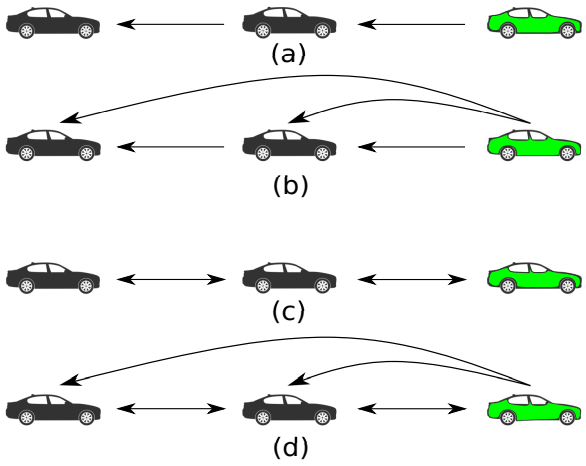


Figure 2.2: Common information flow topologies used in platooning: (a) Predecessor Following; (b) Predecessor-leader Following; (c) Bidirectional; (d) Bidirectional-leader.

is presented in [48] claiming that the best approach to V2V communication is to use the bidirectional-leader approach, denoted as (d) in Figure 2.2, which permits the transmission of data from and to the leader for each node of the platoon along with the path presented in Figure 1.1. This topology increases the platoon stability because the size of the platoon does not affect the response times of the vehicles in case of emergencies; the leader can directly communicate with all the cars thus avoiding the latency involved with propagating messages from the head of the platoon to the back.

Platoon management is also concerned with the formation of the platoons. There are two ways in which they can form; the first involves the vehicle being in the same place while the latter is the scenario we have seen already, that is, vehicles coming together during their trip and joining each other. For the latter, also commonly named as Ad Hoc, in order for a car to join the platoon, it needs to know the $vehicle_{id}$ of the leader. This is achieved, either by getting the relevant info from the special RSU that contains a list of nearby platoons and their destination or by receiving a message from a car that belongs to a platoon, usually the last car; we already saw that the leader can broadcast the status of the platoon, this message also includes the leader's id. After that, a message is sent to the platoon leader and the car awaits the response, with the protocol continuing as already described in Section 2.2.1. For the former, named Planned, the situation is a bit different, as it usually involves heavy duty vehicles [7] making long distance journeys. In this scheme, the trucks can originate from the same location, be destined for the same area or have

a common route along the way. The convoy forms before the trip starts and any other vehicle that wants to enter the formation, after this occurs, needs to wait at highway hubs where truck drivers rest. This entails an evaluation of the waiting time against the cost savings of joining a convoy. In either case, it involves trucks that most likely belong to different companies and thus cooperation is needed, especially for picking the leading vehicle who enjoys the least of the benefits [7].

This last point is not trivial to solve. The driver at the head of the convoy needs to operate his vehicle, in contrast to the followers who can rest, and his position provides him the least gains from the reduced fuel consumption. In [49] they tackled the problem by applying an approach based on game theory in order to choose the leader in an Ad Hoc environment; their research showed that collaborative work, meaning everyone at some point will be a leading vehicle, is beneficial in the long run. Another way to approach this, is by monetary incentives. The leader is compensated by the rest of the platoon by taking some of their cost saving earnings, thus making it worthwhile. A score system has also been investigated, with the vehicle in front decreasing its score while the followers increase it. This guarantees that the same car, and by extension the same driver, is very unlikely to lead a convoy in the near future. Both of these approaches, and a few more, are discussed extensively in [7].

2.2.3 Fault Tolerance

To make the system more robust, and to adhere to the safety requirements in the automotive industry, it is crucial that even in the event of a communication failure the vehicle continues to operate. To compensate for instability and any possible interference that may arise when a beacon is not received (every 100 ms), the car automatically downgrades its operation mode to ACC. In this approach [50], the same timing intervals apply but extra operational states exist with the intention of facilitating a better usage of the available sensors. Range finding is achieved by various means, through the use of Light Detection and Ranging (LIDAR) combined with ultrasonic sensors or, in the case of failure, by only one of them. The same applies for learning the speed of the vehicle ahead. If there is a failure with the specific node, the scheme described in [47] dictates that the car communicates with the leader in order to reach the car in front of it, in contrast to the uni-cast one-way flow we saw earlier. The controllers we discuss later in Section 2.3 do not require this functionality. In addition to using several sensors for redundancy, the leader is responsible for

capturing a snapshot of its environment and passing it to the vehicles at the rear. This way, every car can calculate the probability of the preceding car braking thus avoiding false positives that can lead to excessive breaking and injury. Moreover, each car can calculate a break threat number corresponding to the acceleration, positive or negative, needed to avoid a collision.

All these measurements give rise to the following state space. If everything works as intended the car operates under CACC mode. When there is a network communication failure (missed beacon) the state is changed to the *Fault-Tolerant* state and adjusts the intra-platoon gap without degrading to ACC. From this point, three possibilities exist. In case of a restored communication the gap is again closed. To reach the *Intermediate* state, in which the car brakes, the following must be true. Five beacons must never arrive, or 500 ms elapse without communication, and the acceleration and breaking probability are above and below, respectively, of a threshold. This state exists to adhere to the safety breaking requirements required in ACC operation mode. Howerver, if the rate at which the distance between the car and the preceding vehicle is closing fast, then the state changes from *Intermediate* to *Fail Safe* in order for the cars to avoid a collision. Finally, if the breaking probability of the leader is big enough, the car automatically falls into the *Fail Safe* (from *Fault-Tolerant*) state. Figure 2.3 illustrates the above steps as a potential way in fault tolerance.

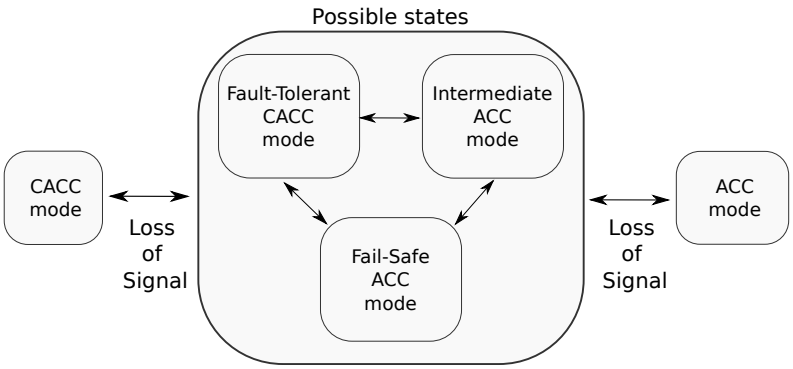


Figure 2.3: Transitional states between CACC and ACC inducing bigger intra-platoon gaps and softer breaking.

2.2.4 Managed Lane Strategies

Typically, the platoon does not travel alone on the road. Vehicles that do not have CACC capabilities cannot communicate with nearby cars and thus are

not aware of their actions, e.g. when a vehicle wants to brake. Additionally, in a highway the platoon may have to change lanes in order to take an exit; all these scenarios are potential problems for the applicability of platooning. The study presented in [21] illustrates the differences in network utilization, traffic safety and throughput when using different kind of lanes on the road. The possible road configurations are the following:

- Regular case: A special heavy-occupancy lane exists on the road. This lane is only used by buses and CACC cars are treated as General Purpose (GP) vehicles.
- Unmanaged lanes: Every vehicle is treated as a GP car and no special privileges exist for the different categories of cars.
- Mixed Managed Lane: This configuration allows cars with CACC capabilities to have priority in the usage of the lane.
- Dedicated lane: This is the case we have already mentioned.
- Access Controlled Dedicated lane: The heavy-occupancy lane is solely used by CACC cars, as above, but entering and exiting the lane is only allowed in certain spots on the road.

The difference in the last two lies on the fact that for cars to enter the left most lane, several lane changes may be needed which could lead to accidents. To mitigate this, the necessary distance needed for these weaving actions was calculated in [51] [52].

The results are pretty clear in regards to better utilizing the resources at hand. When there are less than 30% of CACC enabled vehicles, using dedicated lanes is counterproductive diminishing not only the throughput but also the fuel consumption; which comes against the goal of platooning in the first place. On the other hand, when the market penetration goes beyond the 35% mark, dedicated lanes start to produce better results and, although, fuel consumption is slightly better in the regular dedicated lanes, all the other metrics show that access control can provide a significant boost.

2.2.5 Vehicle Location

Finally, a crucial requirement for not only forming a well-structured platoon but also mitigating related attacks is to have accurate position of the vehicles involved. Several approaches can be used here, for example, each vehicle can perform a Message Consistency Check (MCC), that is, to construct a local

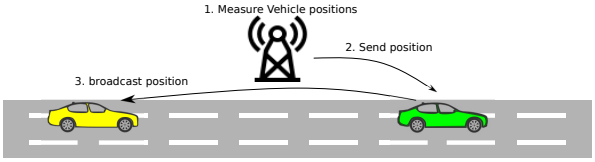


Figure 2.4: Dissemination of vehicle location with the help of a Road Side Unit.

world model based on intercepted messages from its surrounding [53]. In addition, it can use its sensors (e.g. LIDAR) to not only authenticate the cars around it but also verify their existence, to protect itself from nodes that try to create extra traffic in a malicious way. Depending on the attack model, a collaboration between the vehicles can be used, in a privacy preserving fashion, to get an accurate position of a vehicle using oblivious transfer and a triangle localization algorithm as described in [54].

The RSUs can also be utilized to infer the position of vehicles on the road. Figure 2.4 illustrates a protocol that uses both surrounding cars and a RSU to achieve vehicle localization. The protocol is further discussed in [55] but it can be visualized with the following. After the vehicle registers to the RSU (when entering the highway), it receives an estimation of its location as depicted in step 2 of Figure 2.4, which then transmits to its neighbors. Each node is responsible for capturing these messages and storing them after a classification procedure. If it deems that the location provided is not plausible based on a predetermined threshold it can drop the location as unreliable.

2.3 Controllers

So far, we have seen two separate control functionalities. The regular ACC, that is not adequate for platoon formation, and CACC that is an enhancement to the first. For the latter, we include four different implementations to test with our attacks. At this point, it should be mentioned that in autonomous vehicle platoons a very crucial property is that of string stability. That is, a spacing error that starts from a vehicle at index i does not amplify the more vehicles it affects at indexes $j > i$ [56]. All the controllers we outline here try to guarantee this property one way or another, depending on their spacing policy. The controllers we use in our work can be separated in two categories depending on the intra-platoon distance policy they follow. The standard defines them as Constant-Distance Gap (CDG) and Constant-Time Gap (CTG) [45], but in the literature they can also be found with a different name. We choose to use the

latter, namely Constant Vehicle Spacing (CVS) and Constant Time Headway (CTH).

2.3.1 Constant Time Headway

In this spacing policy, the intra-gap of the platoon is linked with the speed with which the vehicles are traveling. This means that the faster the vehicles are moving the bigger the gap between them needs to be to guarantee string stability and safety. They achieve this policy by utilizing a special property called *headway*. A simple yet effective way to think of this is the following. The criterion used for the spacing is the time between the rear bumper of the leading vehicle and the front bumper of the following vehicle to pass a fixed location on the roadway [57]. Because most of the information to enforce this policy can be obtained through the vehicle's own sensors, communication between the platoon members is limited. At first glance, this policy appears to defeat some of the attacks that we discuss later but it has the downside of decreased traffic capacity [58], due to the bigger intra-platoon gaps; because the vehicles cooperate less any sensor error is also more significant.

2.3.2 Constant Vehicle Spacing

On the other hand, CVS, does not depend on the velocity of the vehicle ahead but always try to enforce a constant spacing between the cars. In order to achieve this, each vehicles needs relevant information from the vehicle ahead in order to calculate its own speed and acceleration to guarantee that spacing. This detail makes this policy more robust against sensor errors and increases the traffic capacity [58] but it has the drawback of relying on information from other cars that may be compromised.

Table 2.1: Comparison Between Different Controllers;
Radar Measurements: R V2V Communication: (•) Top precedence: (•)/R

Controller	Policy	Predecessor			Leader			Topology
		d	s	a	p	s	a	
ACC	CTH	R	R					-
PATH	CVS	R	(•)/R	(•)		(•)	(•)	PL Following
Consensus	Both	R			(•)	(•)		PL Following
Flatbed	CVS	R	(•)/R			(•)		PL Following
Ploeg	CTH	R	R	(•)				Predecessor Following

2.3.3 CACC Controllers

All the controllers in this report, including the common ACC and CACC fall into one of the two categories. Table 2.1 provides a very good visual summary of the spacing policy of each controller along with the information it needs to enforce it and how these data are collected; either through sensors or via V2V communication denoted with the symbol (((•))). The required data include distance (d), speed (s), acceleration (a) and position (p). What sets apart the ACC controller, from the rest, is the inability to use wireless communication to share relevant information; this communication permits the reduced spacing without sacrificing vehicle safety.

The PATH controller [59] is the first controller in this list that involves cooperation with other vehicle which is required to form a platoon. It is the default CACC controller used and adheres to the CVS policy in a predecessor-leader following fashion, presented in Figure 2.2. PATH is configured to give precedence to predecessor data obtained through V2V beacons, instead of its own sensors, to improve its accuracy.

The Consensus controller [60] is using a combination of both policies and is able to utilize several topologies. In its default implementation the predecessor-leader-following topology (Figure 2.2) is used which means it only considers beacons that come either from the predecessor or the leader of the platoon. Nonetheless, the algorithm considers the data from all the vehicles to reach a decision for its acceleration and ultimately its speed. The algorithm is able to sustain a stable string by combining the information from the radar and the beacons, even in scenarios where noise would make one or the other unavailable.

The next CVS controller is Flatbed [61] which, in comparison to the PATH algorithm, requires less V2V communicated information. The speed from the leader is necessary for its functionality but it also gives precedence to the predecessor speed when its disseminated, although with a far smaller weight compared to both the PATH controller and the leader's speed. It achieves its policy by emulating an imaginary tow truck and placing all the vehicles on top of it thus fulfilling the CVS requirement. The only information required is that of the tow truck's speed, or to be precise, of the leader's, which is then used to control each vehicle's speed with the simple equation $u_i - V$, where $i \leq PlatoonSize$ and V being the truck's speed. The predecessor's speed, as is the case with PATH, is used to achieve better accuracy compared to the vehicle's sensors.

The last on this list is Ploeg [62] which only accepts the beacons of the

vehicle in front of it and, thus, utilizes a predecessor-following topology; this topology is shown to suffer from string instability the bigger the platoon size is when there are external disturbances [63]. Ploeg also support some form of graceful degradation in case of network errors. When a vehicle does not receive acceleration information from its predecessor through V2V communication, it predicts that value.

2.4 Related Work

2.4.1 Network Jamming

Considering the nature of platooning, any delay in relaying status information can be catastrophic. This emergent property of the system makes it a prime target for network interference, either in the form of constant noise on the network or of more sophisticated attacks. The malicious actor can, for example, initiate an attack when the platoon tries to avoid an obstacle on the road. If he has innate knowledge of the platoon functionalities he can anticipate the network communication between the vehicles and react to those. This way he can hide in plain sight because any installed defensive mechanism is not able to discern him before the attack starts; this kind of intelligent attacks are a good strategy against mechanisms that rely on trust or other long term data.

In that sense, the first question that needs to be answered is where the jammer should be. In their work, Alipour-Fanid et al. [64] tested several positions for the attacker, which in their case was a drone flying above the platoon, concluding that the best position is above the second vehicle; at that position, the error propagation downstream is the biggest. Their results also showed that the *headway* is a crucial factor for a safe CACC operation and should be taken into account when testing CTH controllers.

In a later study, van der Heijden et al. [65] expanded this idea and placed the jamming source inside the formation, making the attacker part of the platoon. In order to simulate a more intelligent attacker they also initiated the attacks when the platoon was accelerating which proved to be the most impactful. Their analysis shows that the most crucial factor is spacing, while speed does not directly influence the results, with the CTH controllers proving to be resilient against this attack. An interesting remark is the fact that the jamming effectiveness increases when the starting time is in the middle of the acceleration process and diminishes when its near the start or the finish.

2.4.2 Data Injection

Data injection is another aspect of network attacks on vehicle platoons and is one that we also considered. Some work has already been done in this area with internal attackers sending erroneous data downstream, specifically the acceleration, the speed and the position of the vehicle with different effects depending on the controller deployed.

In [65], it was shown that position falsification had zero impact on CVS algorithms, while on the other hand, they were susceptible to crashes when a speed falsification attack took place. Furthermore, erroneous acceleration had varying results depending both on the speeds of the platoon and controllers' intra-platoon gaps.

Iorio et al. [66], took their study one step further and incorporated the leader as a potential malicious actor, which the work of [65] assumed as benign. Considering the position of the leader and its influence on many topologies and controllers, this addition is worth exploring. Their work also included the Flatbed algorithm already discussed in Section 2.3 and they made a remark about more intelligent falsification attacks by either combining all the kinematic values of the transmitted data or by steadily increasing those values.

2.4.3 Misbehavior Detection

So far we have only discussed potential ways of attacking the platoon, but mechanisms to avoid such attacks have also been proposed in the literature. Before explaining some of them it is prudent to define and categorize them in order to have a clearer picture of the misbehavior detection space. A detection mechanism is distinguished by the way it treats the data that it analyzes. If the focus is centered on the data, the mechanism is called *Data-Centric*; on the contrary, if the focal point is the node disseminating the data, it is called *Node-Centric* [67].

The former is further separated into mechanisms that rely on plausibility checks or checks based on the data consistency. Plausibility analysis requires a model in which the data can be compared with, in order to verify them. The mechanisms we outline later fall into this category. Their main advantage is that they are always applicable regardless of the presence of colluders, but their utility relies on the accuracy of the underlying model. On the other hand, consistency based mechanisms compare incoming data with previously obtained ones to reach a consensus on their validity. This can also include the collaboration with other vehicles but it has the side effect of requiring a local vehicle majority, otherwise multiple colluders could circumvent it [67]. The

kalman filter scheme we discuss later and evaluate in this work falls under both of these categories. Node-centric mechanisms are also classified into two distinct categories, behavioral and trust based. The first, try to discern if the node is misbehaving based on some pattern, i.e., the frequency of the transmissions or if the node has disseminated the data it should, which is achieved through neighbor monitoring [68]. The latter, are based on the fact that most nodes in a network behave honestly. They use this to create reputation systems, rating a node's behavior, or voting schemes to discern the correctness of the information received. This kind of schemes are valuable because they can simplify the revocation process when a misbehaving node is detected. However, they are susceptible to Sybil attacks where an attacker can create multiple identities to abuse the scheme to either evict honest nodes or amplify his attack.

In [69], several mechanisms are tested, though their effectiveness against the more intelligent attacks, like the steadily increasing speed rate, would be minimal. Some of the mechanisms tested include Simple Speed Check (SSC) and Acceptance Range Threshold (ART) with the first simply making sure that the received speed deviation is inside a certain range while the latter uses the expected reception range to measure the plausibility of the position information received from the predecessor.

Another alternative to misbehavior detection is considered by Garlichs et al. [70]. Their system analyzes every incoming beacon for each vehicle and tries to calculate a trust factor. The more consistent a vehicle is the more trusted it gets, with few errors diminishing that trust faster in order to simulate a real world scenario. Considering the volatile structure of vehicular platoons this alternative has its own drawbacks and, according to their results, it is susceptible to noise when the platoon is accelerating.

In addition to testing the leader as a malicious attacker, the work presented in Section 2.4.2 tries to find a countermeasure to those attacks. It proposes an approach based on Kalman filters, probabilistic structures that use the previous states and incoming data to predict the next state. These filters are potent in environments with noise and are able to correct the drifts created by them [66]. They combine the filter with the fusion of the kinematic properties transmitted, their plausibility model. A visual representation of this is presented in Figure 2.5. At the 40 seconds mark the attack begins. It takes the mechanism a little more than one second (total 14 beacons) to flag the behavior as malicious.

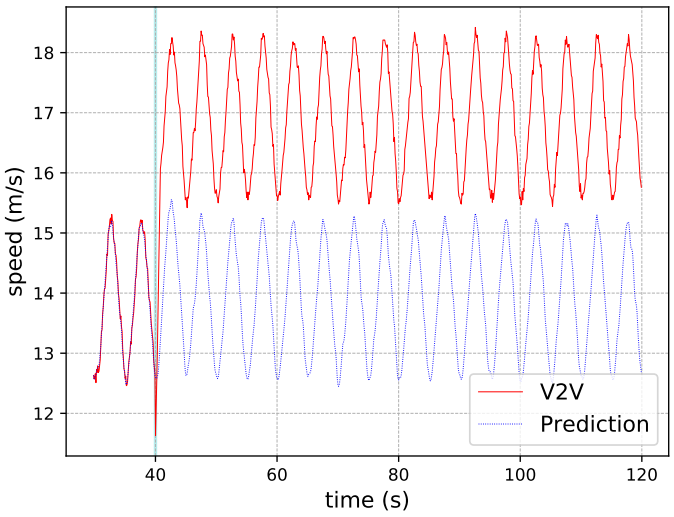


Figure 2.5: Kalman Filter Detection; the attack starts at the 40 seconds mark (vertical cyan line) and is flagged in a bit more than 1 second.

Table 2.2: Related Work Comparison.

Attack	speed variety	malicious leader	maneuver	realistic sensors	limitation
Jamming	no	no	no	no	outside attacker
Jamming & Falsification	yes	no	no	no	single attack time
Falsification	no	yes	no	yes	single attacker only positive falsified data

2.5 Summary

To summarize what we discussed so far in this chapter, Section 2.4 delved deeper into how internal jamming and falsification attacks work and provided some information on their effects to the string stability and the safety of the controllers. It also gave a brief introduction on the way detection mechanisms work and outlined some of them.

To continue further, and make it easier to later compare our work, the relevant data are condensed into the simple Table 2.2. In regards to injection attacks, tests with different leader speeds have been performed for an attacker residing in the convoy but without occupying the first position, which could potentially alter the behavior of certain controllers based on their topologies. On the other hand, experiments done including a compromised leader have only tested falsifying positive values which is unnecessarily limiting.

Chapter 3

System Model and Methodology

In order to achieve a comprehensive and sensible outcome certain decisions had to be made, both in respect to the scope of this thesis and the tools that were used. This Chapter describes these decisions and the methodology followed to make sure that both the data generation and their subsequent analysis is valid. Section 3.1 discusses the assumptions and the adversary model we consider in order for our experiments to be scientifically valid. The next section, Section 3.2, outlines the software engineering aspects of this work and the steps we took to make the resulting software easier to follow and expand. Section 3.3 delves into the tools we chose to use, the overall design of the experiments, what they aim to achieve and how we analyze them to get meaningful results. Finally, in Section 3.4 we provide the reasoning behind the validity of our experiments.

3.1 System Model

3.1.1 Assumptions

We assume that all the vehicles in our tests use the VPKI architecture. Furthermore, each vehicle has already received its set of pseudonyms from the PCA and all the communications are mutually authenticated by using the aforementioned pseudonyms. In regards to the join procedure, the vehicle that wants to enter is already aware of the presence of the platoon and knows the identity of the platoon leader. In our tests, all the vehicles have the same capabilities which include engine power and braking distance, this provides a uniformity in our offensive experiments. Additionally, for this work, all of them are of the same length, which simplifies the distances that cars needed

to create when a join maneuver is approved. All the cars are deployed with sensors capable of measuring the distance from the vehicle in front, deducing the predecessor speed without the need for their input, and are capable of V2V communication.

3.1.2 Adversary Model

In order for the results to be meaningful we need to define an adversarial model with which we can arrow down, but also focus, or work. Firstly, we are primarily concerned with internal attacks, adversaries outside the platoon formation are not considered part of the scope; this means that attacks that rely on identity managements faults or omissions like Sybil [32] or Man In The Middle (MITM) are not considered. This has the implications that the cars accept as valid all the messages coming from the platoon, thus, giving way to falsification attacks that rely on the data of a valid message being erroneous. The compromised vehicle can be either the leader, a member of the platoon or a collaboration between several vehicles. The falsification attacks that they can mount are focused in manipulating the information transmitted through the beacons; mainly the speed, the acceleration and the position of the vehicle. Secondly, the attacker can only interfere with the data send from his car and any other action, like changing the direction of movement, is prohibited. This distinction makes it so that the passengers of the cars may not be aware that their vehicle is misbehaving which allows the attacker to also make the compromised vehicle a victim. Thirdly, we assume that the compromised car cannot change the protocol of the controller or affect the hardware of the vehicle in an way. Consequently, maneuvers and course corrections cannot be interfered with and the sensors reading cannot be altered; this does not exclude the possibility of changing their data before transmitting them. This narrows down the possible attacks that can take place but also makes them more likely to occur because fewer vulnerabilities need to be exposed.

3.2 Implementantion Details

Apart from implementing all the attacks and the defensive mechanisms, we took steps to design code that could easily be extended and modified dynamically before each simulation run. In the following sections we try to showcase some of the changes that perhaps can be designated as the most important. That said, not all of the implementation details are important to

show here. The way the scenarios are executed or how the messages are sent are not included, as they serve as the backbone of the actual code tested.

3.2.1 Injection Attacks

The work done in [65] also provided some code including the jamming attack, which is as simple as "do not receive packet", and the attacks they performed, which for our purposes are not adequate. The jamming attack is meant to simulate the inability of the vehicle to receive a packet and is not concerned with the way this is achieved. We follow the same approach as our goal is to measure the resilience or the susceptibility of the controller under communication failures regardless on how this is achieved. Moreover, we extended the attacks in order to simulate an intelligent attacker. Listing 3.1 shows an example of data falsification made by a smart attacker. He gradually changes the falsified data to avoid mechanisms, such as ART and SSC, but also takes care to combine the other kinematics properties to fool defensive mechanisms that use data fusion. Lines 4–5 demonstrate the former while lines 18–21 showcase the latter for an attack targeting the position sent. In both cases, the falsified value is changing based on a predefined step and is confined between certain limits. When it comes to the smart attack, the focused kinematic property acts as the base from which we relate the other two properties. Let's focus for example on the position. After falsifying the value (line 18) we change accordingly the acceleration and speed based on the kinematic formulas. The step values try to represent a small enough step that can pass undetected as small jitter, but a significant one when it happens over a prolonged time.

```
1  double BaseProtocol::gradualChangeData(double input){
2    input += shiftX;
3    if(shiftX >= lowLimit && shiftX < upLimit){
4      if(fakePos){//position falsification
5        shiftX += 2.5;
6      }else if (fakeSpeed){//speed falsification
7        shiftX += 0.5;
8      }else if (fakeAcc){//acceleration falsification
9        shiftX += 0.05;
10     }
11   }
12
13   return input;
14 }
15
16 vector<double> BaseProtocol::createSmartData(VEHICLE_DATA &data){
17   double interval = 0.01;//in seconds
18   if(fakePos){ //corellate based on falsified position
19     kinematics[0] = gradualChangeData(data.positionX);
20     kinematics[2] = (kinematics[0] - data.positionX - data.speed * interval) * 2 /
        pow(interval, 2);
21     kinematics[1] = data.speed + kinematics[2] * interval;
22   }else if(fakeSpeed){
23     kinematics[1] = gradualChangeData(data.speed);
24     kinematics[2] = (kinematics[1] - data.speed) / interval;
25     kinematics[0] = data.positionX + kinematics[1] * interval + 0.5 * kinematics[2]
         * pow(interval, 2);
```

```
26    }else if(fakeAcc){
27      kinematics[2] = gradualChangeData(data.acceleration);
28      kinematics[0] = data.positionX + data.speed * interval + 0.5 * kinematics[2] *
         pow(interval, 2);
29      kinematics[1] = data.speed + kinematics[2] * interval;
30    }
31    return kinematics;
32  }
```

Listing 3.1: Implementation details of data falsification depending on the chosen attack.

The code in Listing 3.1 depicts the data falsification that a vehicle performs but this is only one step in the whole injection attack process. When a car is designated as malicious through the configuration file, and the time to start an attack has come, it changes the data based on the attack chosen. If no evil act is chosen, the control flow continuous normally. The data are either changed through the code shown above or by the predefined values chosen at startup, thanks to our modular design. The range of changeable data is described in Section 3.2.4 and illustrated in Section 3.3.2. The data, in the intelligent attacks, are falsified in relation to the real properties of the vehicle, otherwise they would appear as garbage to the incoming vehicles or worse be detected by the simplest of safety checks. With the specific data changed, the control flow resumes nominally and the beacon is prepared and sent. From the moment an attack starts, the malicious car continues to transmit the bogus data, each time updated to the current situation, thus ensuring that the formation downstream is continuously affected. In cases where an attack is performed during a maneuver, the intelligent attacker always chooses to start his falsification in specific moments. When a join is performed, the attacker chooses to initiate the attack a few moments, in the context of beacons interval, before the *Joiner* is ready to enter; when an exit is performed, the attacker starts his attack right after the vehicle leaves the platoon. For the former, the attack aims to destabilize both the *JFollower* and the *Joiner* in order to cause the biggest havoc by increasing or decreasing the intra-gap distances, including the distance in which the *Joiner* finally enters. For the latter, the goal is to confuse the immediate follower when he is accelerating to close the newly formed gap. The attacker can perform the above attacks, with precision, because he is aware of all the joining and exit details as a member of the platoon.

3.2.2 Maneuvers

The tools that we use, described later in Section 3.3.1, to perform the attacks and simulate the platoons' functionality lacked any refined maneuver capability. Platooning Extension for Veins (Plexe) included a simple join

scenario for the PATH controller but it proved to be too narrow for our use cases. In that regard, we expanded the maneuver to support dynamic position entry which required cooperation from the formation's vehicle. We also designed the maneuver to function regardless of the controller used which also required the modification of the underlying tools in order to support it, namely the mobility simulator. The code for both the enhanced join and the developed exit maneuver is too big to be presented here but Listing 3.2 depicts the control functionality of the two vehicles involved when a *Joiner* enters at the middle of the platoon.

Lines 5–32 concern the incoming vehicle while lines 33–54 dictate the functionality of the appropriate follower. The *Joiner* tries to close the gap by constantly measuring the distance with the vehicle in front and notifies the leader when this spacing is smaller than the one required, shown at lines 23–24. The distance the *Joiner* needs to calculate is based on the controller the platoon is using to operate. At this distance, the *Joiner* is traveling next to the *JFollower*. At line 24, one can notice that a distinction happens based on the state of the joining vehicle. The control flow of the algorithm reaches this line with two different states but only notifies the leader in one of them; the difference is a simple but important one. The *Joiner* must wait for the *JFollower* to make space in order to join the formation. During that time, he must be able to maintain that position by compensating for any kinematic changes of the platoon and sensors errors that skew the real values one way or another. Lines 47–48 signify that the *JFollower*, reached the distance required for a vehicle to enter in front of it. To achieve this the *JFollower* needs only to change its *spacing* (in meters) if the platoon is using a CVS policy or to increase its *headway* (seconds) value in the case of CTH. In both cases the distance is double the nominal amount plus the vehicle's length. For a CVS policy this corresponds to, in our case, 10 meters, plus the length of the vehicle in meters whereas for the CTH the distance is calculated with the help of the current platoon speed. This way, when the *Joiner* finally changes its lane and enters the platoon the vehicles behind him are already in the correct positions reducing the instability in the formation. The rest of the protocol messages are not depicted here but the explanation of the maneuver has already been discussed in Section 2.2.1. The detailed process, in a step by step fashion, is presented in Appendix A and is accompanied by a descriptive Unified Modeling Language (UML) diagram for easier comprehension.

```
1  void JoinAtPosition::onPlatoonBeacon(const PlatooningBeacon* pb)
2  {
3    //If we are in correct state
4    if (joinManeuverState == JoinManeuverState::J_MOVE_IN_POSITION ||
5    joinManeuverState == JoinManeuverState::J_WAIT_JOIN) {
6      // check correct role
```

```
7       ASSERT(app->getPlatoonRole() == PlatoonRole::JOINER);
8
9       ...
10      //Treat leader beacon
11      ...
12      // if the message comes from the front vehicle
13      int frontPosition = targetPlatoonData->joinIndex - 1;
14      int frontId = targetPlatoonData->newFormation.at(frontPosition);
15      if (pb->getVehicleId() == frontId) {
16        ...
17        //Get front position from radar or beacon
18        //Get my position
19        //Compute distance
20        //Update front vehicle data to close gap/maintain distance
21        ...
22        // Find the correct spacing based on Controller and speed
23        double desiredSpacing = getManeuverSpacing(pb->getSpeed());
24        if (distance < desiredSpacing + 1 && joinManeuverState != JoinManeuverState
      ::J_WAIT_JOIN) {
25          // send move to position response to confirm the parameters
26          MoveToPositionAck* ack = ...
27          //Notify leader, we are next to our position
28          app->sendUnicast(ack, targetPlatoonData->newFormation.at(0));
29          //Change our state to wait for leader confirmation/space to be created.
30          joinManeuverState = JoinManeuverState::J_WAIT_JOIN;
31        }
32      }
33    }else if (app->getPlatoonRole() == PlatoonRole::JFOLLOWER){
34      // The vehicle behind the joiner needs to make space
35      ASSERT(app->getPlatoonRole() == PlatoonRole::JFOLLOWER);
36
37      ...
38      //Treat leader beacon
39      ...
40      if (pb->getVehicleId() == frontId) {
41        ...
42        //Get front position from radar or beacon
43        //Get my position
44        //Compute distance
45        //Update front vehicle data to close gap/maintain distance
46        ...
47        double desiredSpacing = getManeuverSpacing(pb->getSpeed());
48        if (distance < desiredSpacing+1 && distance > desiredSpacing-1) {
49          // notify the leader. Joiner can join the formation in our lane
50          MoveFollowerToPositionAck* fack = ...
51          app->sendUnicast(fack, positionHelper->getLeaderId());
52        }
53      }
54    }
55  }
```

Listing 3.2: The Joinner approaches the platoon and notifies the leader when ready. The JFollower then start making space and finally notifies the leader to allow the Joiner to enter.

Because our attacks may affect the joining vehicle, we decided to create a cutoff mechanism that aborts the maneuver when the positioning of the *Joiner* is not nominal. To elaborate a bit more, lets consider the scenario where a vehicles decides to join at some index in the middle of the platoon. While the gap is created, an attack starts by a preceding car. The *Joiner* tries to compensate by either increasing or decreasing its speed; this can lead the *Joiner* to overshoot the vehicle ahead or increase his distance an amount that positions him several vehicles behind his intended target. If we naively ignore the current position of the *Joiner* and follow through with the protocol, then, the *Joiner* can enter the platoon in a different position, which can prove catastrophic. To illustrate this problem with some relevant plots, we include

some of those experiments in Section 4.4. This mechanism, though, has the unintended effect of restricting the available space that is needed to enter the platoon and can be used by an attacker to effectively deny the completion of join maneuvers. Nonetheless, this is preferable to vehicle collisions. In the case of the Consensus controller, we took an extra step to insure that the space is created in a safe manner. Increasing the headway parameter to the appropriate value, in order to induce double the default spacing plus the vehicle length, has an immediate and abrupt effect. To avoid that, we opted to create the gap in two steps by increasing the headway time appropriately; this way the vehicle's operation is more comfortable. Another interesting implementation detail, in regards to maneuvers, is linked to how Simulation of Urban Mobility (SUMO) handles lane changes. Internally, when a vehicle needs to change its lane it checks if such a move is acceptable; the move is valid if there are no cars in the new lane in the same spot. When that happens, the car is immediately transported without any velocity change or gradual lateral movement. This detail is apparent in the plots detailing the attacks on join maneuvers that we showcase in Chapter 4.

3.2.3 Kalman Filter

The kalman filter mechanism was provided in their work [66] but several modifications had to be made in order for it to work with our design and toolchain. One of them was to port the implementation to a different math library that required the construction of extra test scripts to compare its operation's results to make sure that the it worked as designed.

3.2.4 Replicability

Our design makes it possible that anyone who wants to replicate or extend our work can easily do so by changing the configuration files. These changes can happen before each simulation, thus, permitting their dynamic change. This made it easier to test a large combination of different properties, presented thoroughly in Section 3.3. Some of the configurable values include:

- Operate with or without a defensive mechanism.
- Use realistic sensors or not.
- Starting time for the attacks.
- The attacking node(s).

- The type of the attack and the range of data to inject.
- Perform a smart or gradual attack, choose their focused kinematic property and define their upper and lower limits.
- If and what maneuvers to perform and with which position of the platoon to interact with.

A more detailed illustration of how we achieve this is presented in Appendix B.

3.3 Tools and Measurements

This thesis is concerned with performing network attacks against vehicle platoons. These kinds of formation are not widely used yet, apart from specific highways and test roads [71] [72]. To conduct our research a more suitable approach was chosen, one that is largely followed by the research community. Several tools exists today that can simulate not only the formations, but also their network and the traffic on the road. To that end, in this section we discuss these tools and how we used them to perform our experiments.

3.3.1 Test Environment

Three tools were used to perform the experiments, Plexe [73], SUMO [74] and Objective Modular Network Testbed (OMNeT++). Plexe simulates the platooning environment and it is itself an enhancement of the tool called Veins [75], a framework for running vehicular network simulations. It supports realistic vehicle dynamics and the controllers we use to perform our tests. Its code also includes several scenarios like obstacles on the road and pedestrians crossing which we opted not to utilize as they did not fit with our experiments. We, instead, decided to use the scenarios that introduce speed changes to approximate real world jittering. Before choosing Plexe as our tool of choice, we investigated another tool called Vehicular Network Open Simulator (VENTOS) [22] which is a C++ based simulator for studying vehicular traffic flows. VENTOS has the advantage of providing several platoon maneuvers, such as merge and split, but is also limited to a single CACC controller, PATH; this is the main reason we opted to continue with Plexe.

In order for Plexe, and VENTOS, to function it requires the other two tools on the list. OMNeT++ is a discrete event simulator used to model communication networks of distributed systems and microprocessors [76]. It

is the basic block for both Veins and, subsequently, Plexe and lies at the heart of our toolchain. It also functions as the bridge between the aforementioned network simulators and SUMO; the traffic simulator which is responsible for operating the vehicles, creating the roads, handling the vehicle sensors and implementing the various objects used by the scenarios of Plexe.

Most of the engineering aspect of this work is focused on Plexe as it represents a higher level of abstraction but certain changes were made in SUMO in regards to the available controllers in order to make the maneuvers possible. All three tools, are implemented in C++ and this is also the language of choice for our work.

3.3.2 Simulations

Table 3.1: The Configuration Parameters used by each experiment; Gradual and Smart attack values correspond to upper and lower limits instead of absolute values.

Configuration	Attack Values	Attack Position	Maneuver
JammingDetail	[30-35] s	1	[no,Join,Exit]
PosInjectionAttack	[3, 5, 7, 9, 11] m	[0,2]	[no,Join,Exit]
SpeedInjectionAttack	[-50, 0, 50, 100, 150] km/h	[0,2]	[no,Join,Exit]
AccInjectionAttack	[-30, -10, 0, 10, 30] m/s^2	[0,2]	[no,Join,Exit]
GradualPosFalsificationAttack	[-10,40] m	[0,2]	[no,Join,Exit]
GradualSpeedFalsificationAttack	[-10,17] m/s	[0,2]	[no,Join,Exit]
GradualAccFalsificationAttack	[-10,10] m/s^2	[0,2]	[no,Join,Exit]
SmartPosFalsificationAttack	[-10,10] m	[0,2]	[no,Join,Exit]
SmartSpeedFalsificationAttack	[-10,10] m/s	[0,2]	[no,Join,Exit]
SmartAccFalsificationAttack	[-10,10] m/s^2	[0,2]	[no,Join,Exit]

Section 3.2.4, touched briefly on this but here we elaborate more on the multitude of tests we were able to perform, thanks to both ours and of the tools modularity. Common to all our experiments is the sinusoidal behavior of the platoon leader's speed. This change in speed gives also rise to some interesting results that we see and discuss later in Chapter 4. For each of the attacks, several different data were tested to test the controllers handling. In our simulations vehicles have a length of 4 meters and thus position falsifications start with very small deviations and reaches the length of three cars (gradual and smart attacks) or more (simple position). For the simple position attacks, the falsification is at first minimal but it increases the longer the test is running reaching huge values. The range of values is crucial especially for the CVSs policy controllers which are tested with a space of 5 meters. The speeds,

on the other hand, try to simulate scenarios ranging from a hard brake to smaller variations to extreme speedups. The gradual and smart test limits of -10 and +10 are put in such a way to make sure that the attack could be a benign change of pace. There are two exceptions, one is the gradual position attack that reaches +40 for a total change of 50 meters; this is because, Ploeg and Consensus (CTH) have larger gaps which make a maximum +10 false displacement ineffective. The other is the gradual speed falsification which reaches the value of +17 m/s. This value may seem strange but corresponds to roughly 60 km/h which is not an unheard increase, e.g. when someone enters the highway. For the smart position falsification, the falsified value is combined with the other properties and so bigger values do not necessarily change the outcome. Nonetheless, we chose to limit the range of the smart attacks in order to approximate an attacker that wants fast results and also to make them appear as benign as possible. Each configuration, for each data falsified, was run for two different attackers, the leader and a follower denoted in Table 3.1 with their index, 0 and 2 respectively. In the jamming attack, the actor is residing at the head of the second vehicle, position 1 in the platoon, as described in Section 2.4.1. The jammer effectively stops the leader messages and thus we can also place him at the position of the leader. Each of these experiments were performed for a platoon that did not perform any maneuver, for one that executed a Join and one that decreased its size by performing an Exit maneuver.

Table 3.2: The internal properties used by each platoon during the experiments; CTH controllers use a time value (headway) for their spacing.

Property	Value
Controller	PATH, Ploeg, Consensus, Flatbed
Spacing	5m, 0.5s, 0.8s, 5m
Platoon Length	6 / 7 (Join)
Leader speed	50, 80, 100, 150 kmph
Sensor Errors	$\epsilon_p^{V2V} = 1$ m, $\epsilon_s^{V2V} = 0.1$ m/s, $\epsilon_a^{V2V} = 0.01$ m/s^2 $\epsilon_p^{RAD} = 0.1$ m, $\epsilon_s^{RAD} = 0.1$ m/s

This table, though, only shows half the picture. Table 3.2 illustrates the properties used by our platoons during the tests. In this table, the leader speed is the value that changes based on each run while the others are specific for each controller. The spacing and headway values used are taken directly from their default implementation in Plexe and we conducted most of our tests

with the use of realistic sensors by simulating their small inaccuracies; we introduce small deviations picked uniformly at random for all the sensors. ϵ_{RAD} corresponds to an error in the radar sensor whereas ϵ_{V2V} is related to network communication. The leader speed values are chosen in such a way in order for the experiments to provide information applicable in real word scenarios, from low up to highway speeds. We keep the platoon size at this value for two reasons, first to avoid extending the SUMO implementation of the Consensus controller, that supports up to 8 cars, and because based on recent research bigger platoon sizes are less attractive [7].

Table 3.3: The simulation parameters used to perform our tests.

General Parameters	Value
Simulation length	120s
Warm-Up period	30s
Area Size	5 KM x 50 M (4 lanes)
Repetitions	10
Carrier Frequency	5.89 GHz
Max TX Power	100mW
Physical Layer Bitrate	6Mbps
Sensitivity	-94dBm
Thermal Noise	-95dBm
Physical Layer Propagation Delay	true
Network Layer Propagation Delay	true

Table 3.3 summarizes the general parameters used in all our tests that do not change between each experiment. We run each simulation for a specific amount of time (120s) and we included a warm-up period (30s). If there is a crash we choose to stop the simulation because the goal for the attacker is achieved. Also, we included the warm-up period to eliminate any inaccuracy and errors induced by our starting parameters. The network frequency used is based on the standards defined by ETSI and we use the default parameters in regards to the presence of thermal noise on the environment. We decided to induce delays in both the Network and the Physical Layer. The delays are generated based on a seed that changes in each repetition to provide uniqueness. Finally, we perform each experiment ten times to have some form of statistical significance in our results which is something that can be further expanded.

It is easy to realize how all these can lead to a huge number of tests even if run only once, showed in Table 3.4. We handled the huge load of data by

writing tools to analyze and plot them together.

Table 3.4: Number of Different Experiments.

Configuration	#tests
JammingDetail	44
Simple injections	720
Gradual injections	48
Smart injections	48
-	908

3.3.3 Data Analysis Technique

Considering the number of simulation run, we decided to represent them in specific ways in order to not diminish our work and to also provide to the reader a valuable lesson. Research done on road accidents stipulates that the most contributing factors to injuries and fatalities is the abrupt change of speed at the moment of a collision [26]. In that regard, we opted to use this metric when a crash occurs. For the tests that the controllers managed to avoid a collision we decided to plot a number of different graphs depicting car properties like speed, acceleration and distance from each predecessor; we believe these would be the more interesting things to observe, as they represent the handling of an attack by the controller. To quantify the string instability created by our injections and jamming events, we plotted the difference between the nominal intra-platoon gaps and the gaps that emerged during the attacks. Table 3.5 shows all the choices we made and what we aim to show based on each plot.

Table 3.5: The produced results by our data analysis based on the depicted property.

Depict	Plots
Platoon properties	Position, Speed, Acceleration, Distance, Controller Acceleration
String stability	Intra-platoon gap differences
Collisions	Crasher's ΔV

3.3.4 Analysis Tools

The previous section talked about our analysis design, here, we give some illustration on how we achieved the end results that we aimed for. We decided

to use Python to create our analysis tools, as we deemed it to be easy to handle and handy in creating graphical plots in all short of orientations. In that decision, also, helped the fact that [65] also provided some basic data gathering tool which we decided to use as a basis for our own. The first tool, illustrated in Listing 3.3 bellow is responsible for gathering the data from the logs generated by OMNeT++. Lines 5–14 iterate the input lines and categorize the vectors that match a desirable datatype, which in our case corresponds to the platoon properties shown in Table 3.5. Having done that, the control flow moves to read the data from the rest of the log file, matching the vector numbers and assigning them to a list that is used to plot the platoon properties and also be consumed by the more advanced scripts that we show below. We can see at line 30 that we do not only store the value of the given property but also the timestamp associated with it.

```
1  for line in content:
2    ...
3    #Ignore headers and run parameters
4    ...
5    elif vectorHeader:
6      #parse this vector header
7      data = line.split()
8      vectorID = int(data[1])
9      module = data[2].split('[')[1]
10     carID = int(module.split(']')[0])
11     dataType=data[3]
12     if dataType in desiredDataTypes:
13       vectorToID[vectorID]=carID
14       vectorToType[vectorID]=dataType
15     continue #done parsing vector header
16
17   #parsing data
18   if not paramHeader and not vectorHeader and line != '\n':
19     data = line.split('\t')
20     vectorID = int(data[0])
21     #data[1] is event ID
22     timeStamp = float(data[2])
23     value = float(data[3])
24     if vectorID not in vectorToID:
25       continue #ignore this data.
26     carID = vectorToID[vectorID]
27     dataType = vectorToType[vectorID]
28     if dataType == 'distance' and value == -1 and standalone == 1:
29       value = 0
30     carList[carID][dataType].append((timeStamp, value))
```

Listing 3.3: How to read the logged vectors measuring among others distance, speed and accleration and assign them to vehicles before plotting.

This time value is important for two reasons, firstly, it is an easy way to find out if there was an accident during the simulation and secondly, it is used by our crash analysis tools, shown partly in Listing 3.4, to measure the spacing error of the vehicles which assumes that the times for all the datatypes are the same. If an accident is detected, the tool measures the crash impact in terms of ΔV shown in line 13 and if there is none, it measures the spacing error depending on the controller used. Lines 22–29 depict that in a simplified manner. It should be mentioned here that our tools are capable of measuring a multitude of metrics, e.g., driver comfort through the average spacing error, but we chose

to depict and include in the thesis the most precarious of the results, which is injuries and instability.

```
if accident:
  for v in range(len(carData)):
    if carData[v]['distance'][-1][1] == -1:
      crashingVehicle = v
    ...
    #Parser error handling
    ...
    #Different predecessor
    if exitManeuver and crashingVehicle == 4:
      carShift = 2
    else:
      carShift = 1
    crashImpact = abs(carData[crashingVehicle]['speed'][-1][1] - carData[
   crashingVehicle-carShift]['speed'][-1][1])
    ...
    #Values error handling
    ...
else:
  for v in range(len(carData)):
    for (timestep, value) in carData[v]['distance']:
      #check times are equal
      desiredSpacing = -1
      if controller == '1':
        desiredSpacing = spacing
      elif controller == '2':
        desiredSpacing = 4 + 0.5 * carData[v]['speed'][k][1]
      elif controller == '3':
        desiredSpacing = 4 + 0.8 * carData[v]['speed'][k][1]
      elif controller == '4':
        desiredSpacing = spacing
```

Listing 3.4: Pinpoints the crashing vehicle and measures the platoon instability through distance deviations.

3.4 Data Validity and Reliability

It is easy sometimes to forget that the simulations we run may produce results because of bugs. In this work, we took extra care to make sure that our data are both valid and reliable across all the tests. Apart from meticulously examining the code and running small scale tests to ensure that, we took some extra precautions that we can classify them in the following way:

- Previous work comparison
- Test repetition

Because there was already some research done in our area, we decided to perform tests already present in the literature giving us similar results. This gave us the confidence that our solution works as intended; to further this even more, we found and used simulation logs from those works and analyzed them with our scripts producing akin outcomes, thus, validating our parsing tools. The same also applies for the kalman filter implementation as already mentioned in Section 3.2.3. To guarantee consistency in our tests, we

configured OMNeT++ to run our experiments multiple times using each time the same data as presented in Tables 3.1 and 3.2 (same randomization seed respectively) as an assurance that our code is consistent over a large number of runs.

Chapter 4

Quantitative and Qualitative Analysis

4.1 Metrics

Our goal is to showcase both the platoon instability and the collisions that occur, thus we bundle the experiment results together. In that regard, each plot is separated into two rows and four columns. The different columns represent the distinct movement speeds of the platoon while the rows depict the ΔV when a crash occurs, on top, and the maximum spacing error induced on the bottom. On the bottom row, the correct spacing is presented as a background shade to make the reading of the plot easier. One can notice that the Ploeg and Consensus controllers may have their spacing error beside the default point, this is caused by the jittering of their algorithms and our choice to include the minimum distance they report while the default values correspond to the average. As a reminder, the *Joiner* and the *JFollower* correspond to the vehicle that wants to enter the platoon (in position 3, starting from 0) and the vehicle currently residing in that space respectively; the latter needs to increase the gap to make space for the former.

For the plots that demonstrate the intra-platoon gaps, each vehicle reports the distance from its predecessor, reaching a value of zero means the car crashed with the front vehicle. The leader is always showing zero distance because there are no other cars in front of him on the road; the only way to be involved in a collision is for its immediate follower to bump him from the rear. At the end, we provide several tables that compare the resilience and the vulnerabilities of each controller. Finally, we present a single table that details the potency of all the attacks from the perspective of the attacker.

4.2 Results

4.2.1 Results Hypothesis

Before jumping directly into the figures and analyzing our results we make certain hypotheses in order to have a framework within which we can gain some useful insights.

The first thing that we expect to see is an inefficient position falsification attack when it aims at disrupting formations operating under CVS controllers. Previous research showed that they are not susceptible to such false data although in contrast, CTH, can be affected and that is a knowledge an intelligent attacker can use to his advantage. Additionally, acceleration should affect only PATH and Ploeg because they are the two that use it to achieve their policy; we expect speed to be the most potent, all the controllers except Ploeg use it one way or another.

Our next hypothesis is the resilience of the Ploeg controller to several of our attacks. Ploeg receives through V2V communication only the acceleration of its predecessor and everything else is computed based on its own radar; thus we anticipate some interesting results on attacks targeting this kinematic property. An attacker in that case, has to position himself directly in front of his victim; if the goal is to make a particular vehicle crash. This distinction poses another limit to a malicious actor, attacks on the platoon have to be carefully crafted otherwise he also risks of bringing his own demise, considering his proximity to the affected car.

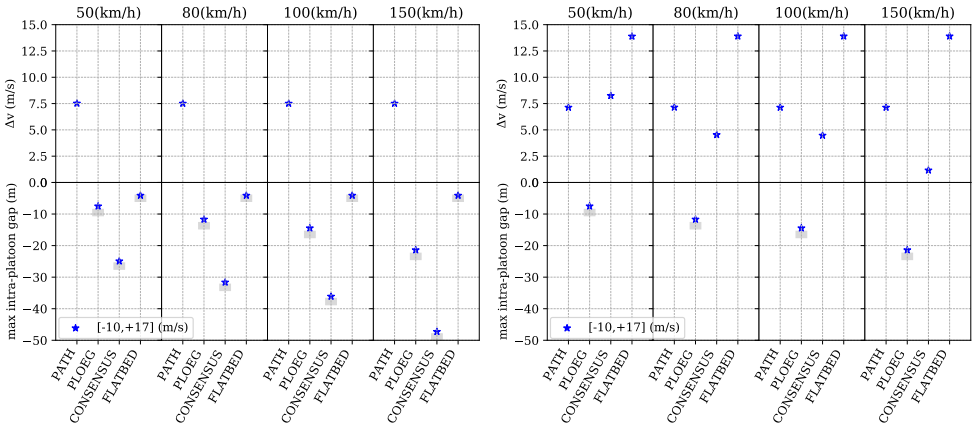
The Flatbed controller poses an interesting case for two reasons. Firstly, it is not extensively tested in these kind of scenarios and secondly, it relies heavily on the leader's transmitted speed. The former, gives us a great opportunity to do a thorough work on examining it in order to discover ways to disrupt its nominal operations. For the latter, considering our attacker can be position at the head of the platoon and the particularities of this scheme, CVS is achieved through a virtual tow truck, the results can be though-provoking. A detail that we need to consider here is that the platoon head enjoys the least benefits in a convoy, making any choice to attack a Flatbed controlled platoon a complex decision that reaches outside the scope of this thesis work.

A central piece of our work is the different effects that the attacks performed by the leader have on the platoon. Except from the Ploeg controller, all the others utilize the information received by the leader to operate and thus we expect them to behave worse under injection attacks. This detail is significant because the leader communicates with all the vehicles of the convoy and is

able to affect all of them at the same time, inducing not only false data through V2V but also by changing the behavior of each vehicle's predecessor.

Because in our tests we aim to see not only crashes but also platoon instability, we expect to see a correlation between the speed of the platoon and the error in the intra-platoon gaps regardless of the controllers involved. For example, we anticipate that CTH controllers, who rely on the vehicle speed to determine the gaps, to have bigger spacing discrepancies the faster the cars are moving. The results here can give a very valuable insight on the best combination of falsified value and platoon speed; thus encompassing convoys that operate in multiple scenarios like freeways, highways or smaller city roads.

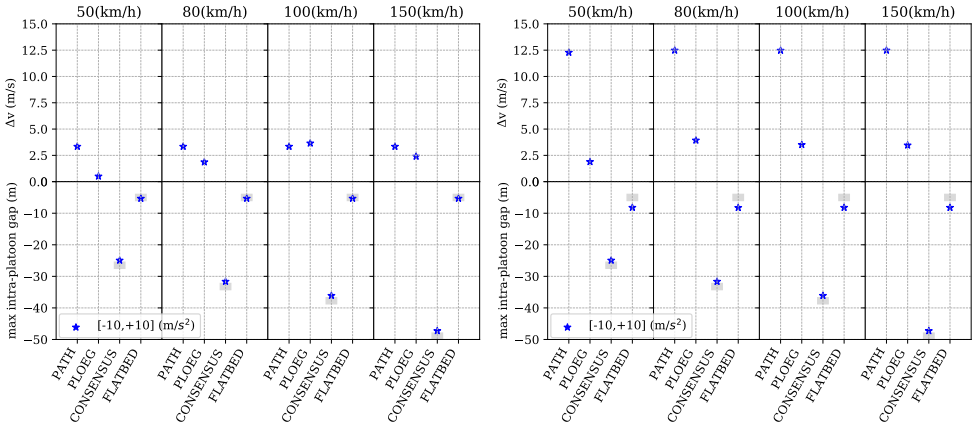
4.2.2 Intelligent Attacks



(a) Malicious Follower (b) Malicious Leader

Figure 4.1: Gradual Speed Falsification Comparison.

We begin by investigating the different responses of the controllers when faced with a gradually changing injection attack. The first set, Figure 4.1, illustrates the effects of speed falsification. PATH crashes for all tests, regardless of the platoon's speed or the attacker's position, with the same severity. Flatbed, on the other hand, is only affected when the attack is performed by the leader; the severity of the crash is significant at approximately 50 km/h. In the same manner, Consensus is affected only by a leader's attack as expected. The severity is following a downwards trend because higher speed correspond to bigger intra-platoon gaps. At the same time, the falsified value reaches a limit at which point it does not increase further. This results

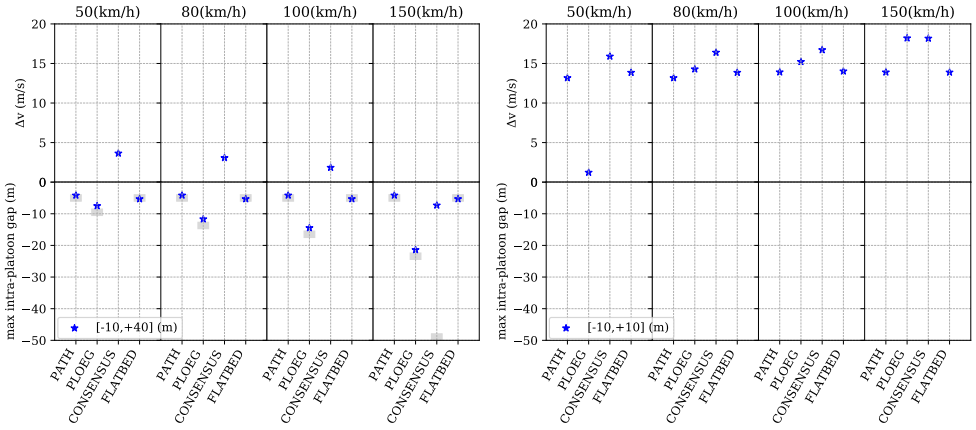


(a) Malicious Follower (b) Malicious Leader

Figure 4.2: Gradual Acceleration Falsification Comparison.

in the vehicle having more time to collide but with a falsified speed that is proportionally smaller to the current speed. Ploeg is resilient regardless of the actor's position as the only data he takes into account is the acceleration of its predecessor. A difference in the collision severity is seen when the falsified data corresponds to the vehicle's acceleration. In Figure 4.2, PATH crashes mildly for the follower and more substantially, with a ΔV of 45 km/h, for the leader. This happens because the attacks result in a different victim. In the first case, the victim brakes colliding with his follower while in the leader's attack, the crash happens due to the second phase of the attack, which involves positive acceleration. A difference can be also spotted for Ploeg. At the lowest speed the attack is less severe for the same reason as PATH. Consensus is unaffected as expected while Flatbed compensates by increasing his intra-platoon gaps. The different collision severity for both PATH and Ploeg is interesting to delve into. It may seem that the more ΔV is better, which usually is, but in this case the lower values correspond to better attacks. The colliding vehicles are not the immediate victim and the attacker but the victim with its own follower. This is a very significant detail because the attacker is not involved in the crash. Later in Section 4.3 we delve specifically into this in order to find the best attacks possible.

To better understand the subtle differences of the attacks, Figure 4.3 provides a comparison between gradual position and the smart position falsification attack. Simply changing the position values is not enough to confuse the controllers, regardless of the attacker's position, except for Consensus which requires the positional value in its control algorithm. The

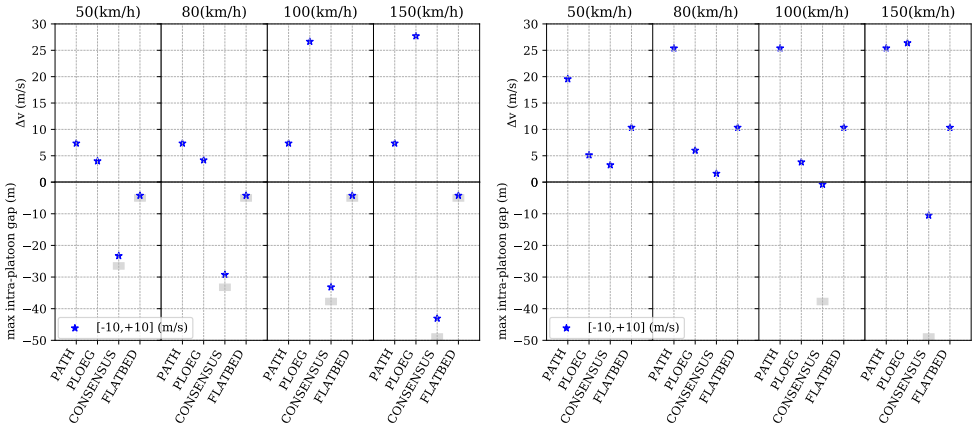


(a) Gradual Position Falsification by Leader (b) Smart Position Falsification by Leader

Figure 4.3: Intelligent Position Falsification Comparison.

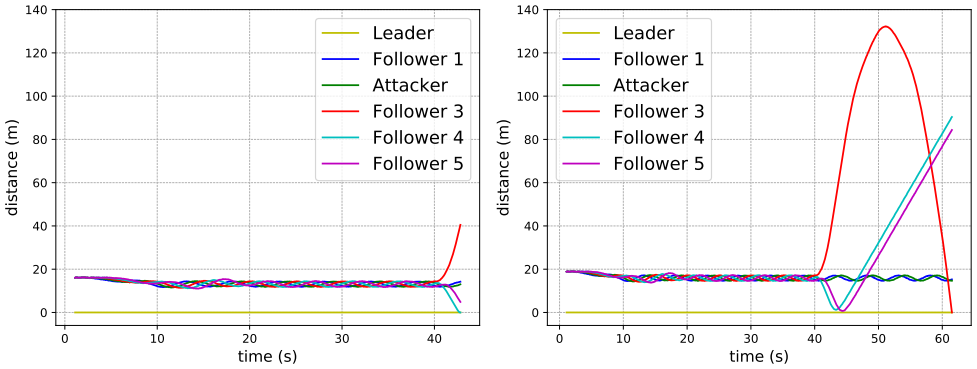
smart attack, though, which also changes the speed and the acceleration, is proving to be very potent, with all the controllers crashing. Collisions that involve the PATH and Flatbed controllers are not affected by the different speeds; this is due to their ingra-platoon gap policy, the spacing is always the same. Ploeg and Consensus, though, have slight increases in the severity due to the increased gaps. An interesting result here is the crash that happens for Ploeg at its lowest speed. The severity is low because the crash involves vehicles downstream and not the immediate follower of the leader.

Figure 4.4, illustrates a combined kinematic properties attack targeting the speed data. The insights gained here are substantial because the results differ based on the position of the malicious actor. PATH fared substantially worse when attacked by the leader and slightly worse, compared to the gradual speed falsification, when attacked by a follower. The smart attack reaches smaller speeds, compared to the gradual one, but the attack also involves a change in the acceleration value which PATH considers, hence the increased severity. The overall severity for PATH hovers between 70 and 90 km/h which is substantial. Ploeg also crashes with this attack. It crashed regardless of platoon speed and attacker position with one interesting detail. At high speeds, the severity is extreme, with a ΔV of around 100 km/h which is comparable to a highway collision. This discrepancy is expanded in Figure 4.5, discussed further down. Flatbed is once again susceptible to attacks originating from the leader of the platoon. This is expected, considering that Flatbed relies on the speed of the platoon leader to conform to his CVS policy. For the follower's attack, the controller is slightly destabilized. Consensus reacts to the follower's



(a) Follower Attacks (b) Leader Attacks

Figure 4.4: Smart Speed Falsification.



(a) Traveling at 80km/h (b) Traveling at 100km/h

Figure 4.5: Follower Smart Speed Falsification: Ploeg Comparison.

attack with a small decrease for the distance between the attacker and his victim. However, when the attack is perpetrated by the leader the lower speeds result to collisions. The result is directly correlated with the bigger intra-platoon distances that occur for the higher speeds. As promised, Figure 4.5 compares the the same attack but for different speeds. For the lesser platoon speed, the attack succeeds by crashing *Follower 4* with the decrease portion of the attack; the victim tried to compensate to the new input by decreasing its speed rapidly, causing his follower to collide with him. On the contrary, at the highest speed, the victim's followers managed to avoid crashing to their predecessor and the attack continued to the increased falsified data part. This lead, the immediate victim crashing on the attacker. This is a valuable

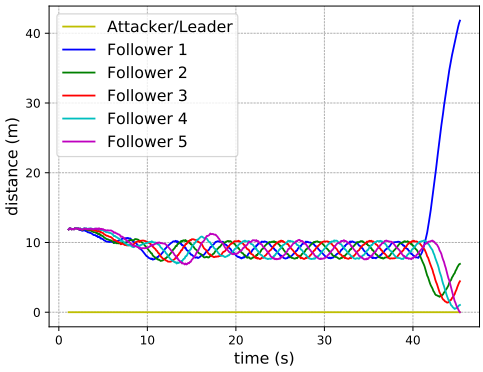
distinction because, most likely, the malicious actor wants to preserve the compromised car.

The smart acceleration effects are shown in Figure 4.6, along with two plots showcasing the crashes of Ploeg and Flatbed respectively. As we can see, Consensus is destabilized with a decrease in the distances between the cars; this is due to the also changed values of position and speed which it considers. As expected, the attack had no effect when initiated by the follower. Flatbed has similar behavior when attacked by a follower, but for the leader's attack it barely crashes. This happens because the vehicles close the distance very slowly until they collide (Figure 4.6.b). PATH and Ploeg have similar results to the gradual acceleration attack. From all our attacks, the smart acceleration attack provides the least upgrade in terms of severity and difference from a simpler gradual attack. This is due to the small values we impose on the attacks; a small change in the acceleration of the vehicle has a very small change in the position and speed properties. Nonetheless, the smart acceleration is more likely to pass undetected in comparison to the gradual attack.

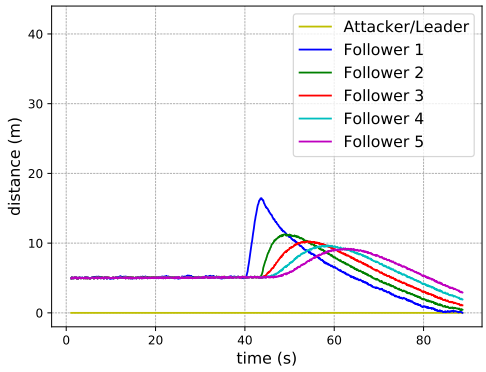
The last proposed attack is presented in Figure 4.7 where we show the smart position falsification. We already presented the leader attack in Figure 4.3. Here, we compare the same attack for a different position in the platoon. PATH and Flatbed are considerably safer, although they still crash. This difference is attributed to the different victims; in the case of the follower, the victims decelerate because of the attack leading to a crash downstream. Ploeg is behaving similarly with the leader's attack but this happens also for the lowest speed. In this case, the crash is avoided at the first stage but happens in the second when the vehicle is increasing its speed thus giving similar crash severity.

4.2.3 Attacks on Exit

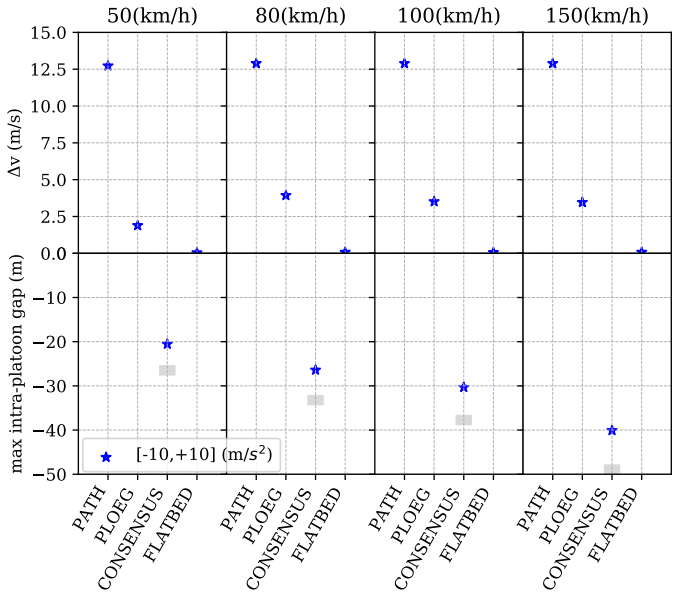
Here, the above attacks, along with the common ones, are used in conjunction with an exit maneuver and begin the moment the attacker notices the operation; when the vehicle changes lane. In the acceleration falsification scenario, Consensus and Flatbed are completely unaffected, while PATH and Ploeg produce different results based on the falsified values. In Figure 4.8 we can observe that PATH crashes for all the induced accelerations except for the zero one. In the common scenario the extreme negative acceleration provides the best results in terms of severity, whereas during the exit maneuver the positive values are more potent. This discrepancy is due to the increased space that



(a) Intra-platoon Gaps for Ploeg



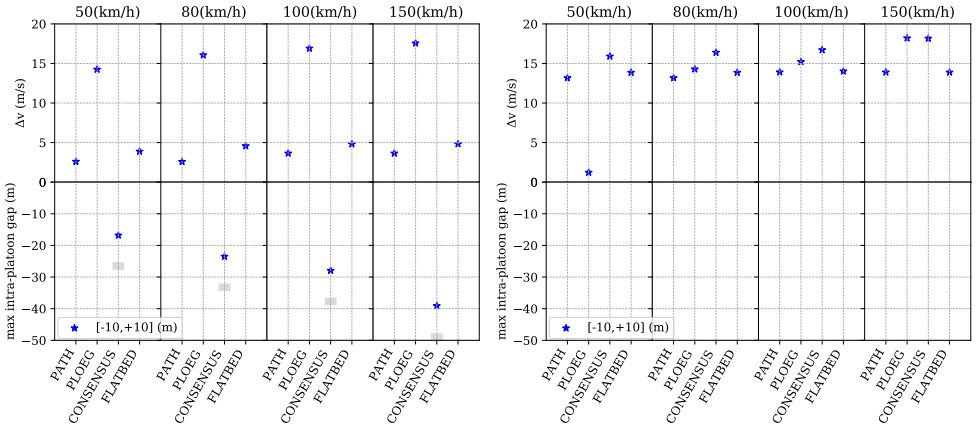
(b) Intra-platoon Gaps for Flatbed



(c) Controllers Comparison

Figure 4.6: Leader Smart Acceleration Falsification Comparison.

exists between the victim and the attacker, thus giving the victim ample time to increase its speed. In the case of Ploeg, the results need further discussion. The small negative acceleration causes significant instability to the platoon except for the common scenario, at the lowest speed. The collision happens due to the breaking of the vehicles between the tail (third vehicle downstream) and its predecessor. In the exit scenario, this does not happen. The immediate victim has a small window in which he accelerates, thus closing the distance, before breaking in response to the falsified data. This small change causes his



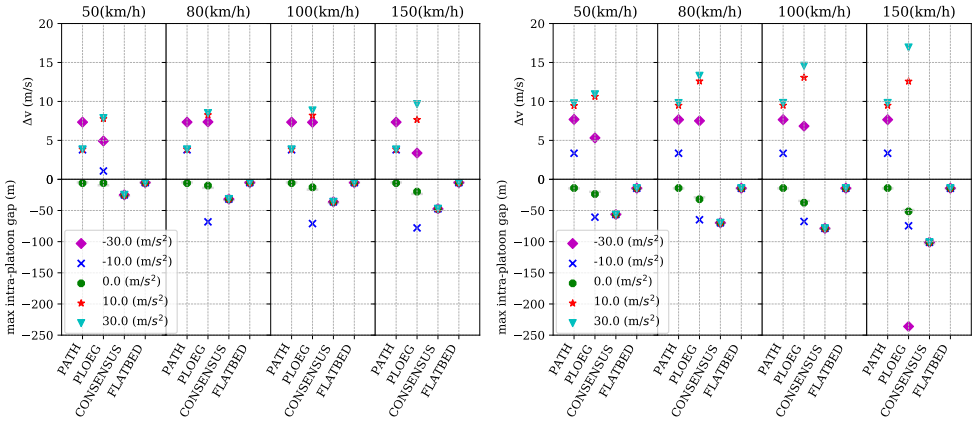
(a) Follower Attacks

(b) Leader Attacks

Figure 4.7: Smart Position Falsification.

follower to better mitigate the decrease in space leading to bigger, in relation, intra-platoon distances. We should point out that because the platoon consists of six cars, in the exit scenario the tail is the second vehicle downstream. This means that the attack may not be avoidable if the platoon is bigger. Nonetheless, we observe bigger distances between the victims during the exit scenario. Moving to the rest of the values, the positive falsifications are more potent in both scenarios. For the exit maneuver, the increased severity is due to the increased intra-platoon gaps. Finally, we can see that the extreme negative acceleration produces an interesting case, both for the common scenario and the exit maneuver. For the former, the attack is more potent for the 80 and 100 km/h platoon speeds. This difference is directly correlated with the distances between the vehicles. In the slowest case (50 km/h) the distances are the smallest, so the gap can be closed faster, but the speed is also lower, while in the highest speed the opposite is true. Thus, an attacker needs to consider the correlation between the speed and the resulting distances when attacking Ploeg. For the exit scenario, the increased distances proved crucial for the vehicles to avoid the crash; this led the victim to increase its gap to 250 meters before stabilizing at 175 meters from the attacker. This distance effectively means the dissolution of the platoon.

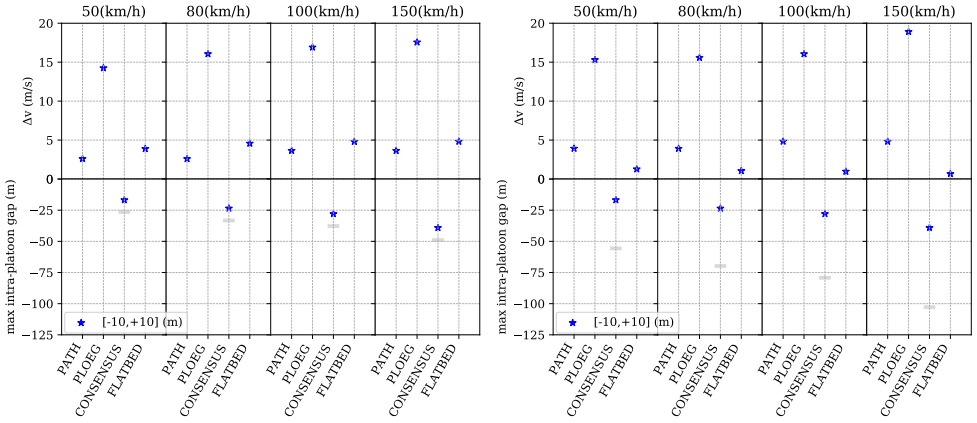
When we attacked the maneuver with a smart position falsification attack, shown in Figure 4.9, the results were almost identical. In some cases the severity dropped, like in Flatbed, whereas for Ploeg it increased by a few km/h. Vehicles operating with the Flatbed controller crashed by extremely breaking, in the common scenario, while they avoided this collision during the



(a) Falsification during a Common Scenario

(b) Falsification during an Exit Maneuver platoon

Figure 4.8: Acceleration Falsification Comparison During Exit Maneuver.



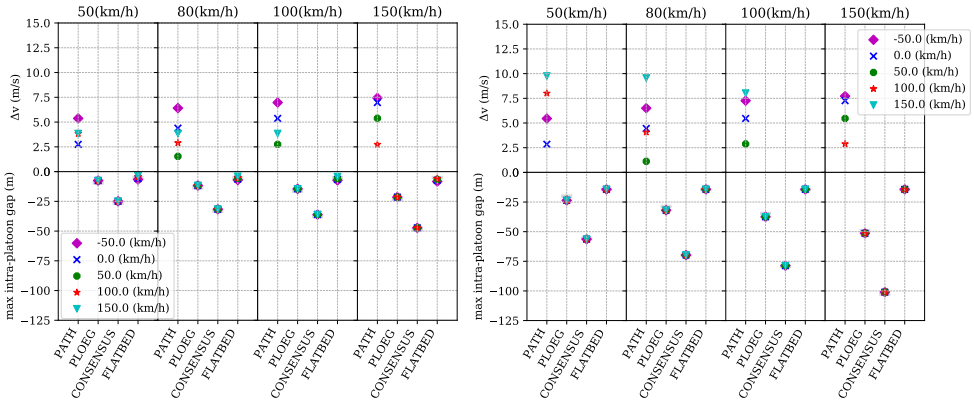
(a) Falsification during a Common Scenario

(b) Falsification during an Exit Maneuver

Figure 4.9: Smart Position Falsification Comparison for Exit Maneuver.

exit procedure; this is due to the increase of the immediate victim acceleration. This increase before the attack proved crucial, making the vehicles avoid the crash and the immediate victim to crash on the attacker during the second phase of the attack. One should not get confused by the Consensus default spacing. The spacing is meant to show the biggest intra-platoon gap during the test; in this case, it includes the exit (double the distance). The instability is identical to the common scenario.

The last attack we will see, in regards to the exit maneuver, is the speed



(a) Falsification during a Common Scenario (b) Falsification during an Exit Maneuver

Figure 4.10: Speed Falsification Comparison for Exit Maneuver.

falsification. The rest of the attacks produced similar results and we do not present them for the sake of brevity. We can see in Figure 4.10 that the attack only destabilizes, slightly, the Flatbed controller and causes PATH to crash. Ploeg and Consensus are unaffected regardless of the scenario. The interesting observation here is that the relatively negative values, for the common scenario, produced better results in terms of severity the higher the platoon speed was. The positive attacks produced the same results regardless. We need, though, to keep in mind that a negative speed is not possible and thus the most severe of the attack is most likely to be flagged by the simplest plausibility check mechanism. In the exit scenario, the same pattern persisted but with a crucial difference. Because the gap in front of the victim was bigger, the extreme positive falsified speed produced more potent attacks. This severity though diminished the faster the platoon was traveling; that is due to the reduced absolute difference between the moving pace and the attack value.

4.2.4 Attacks on Join

The attacks we performed on join gave us a lot of insights on how the controllers work and what is necessary for such a maneuver to work nominally. In order not to distract the reader from the current attacks, we present some of the pitfalls we solved in Section 4.4. An interesting result we observed, is the fact that the controller used by the joiner plays an important role in the maneuver. In Figure 4.11 we present what happens during a gradual speed falsification during the join maneuver. The figure depicts the absolute position

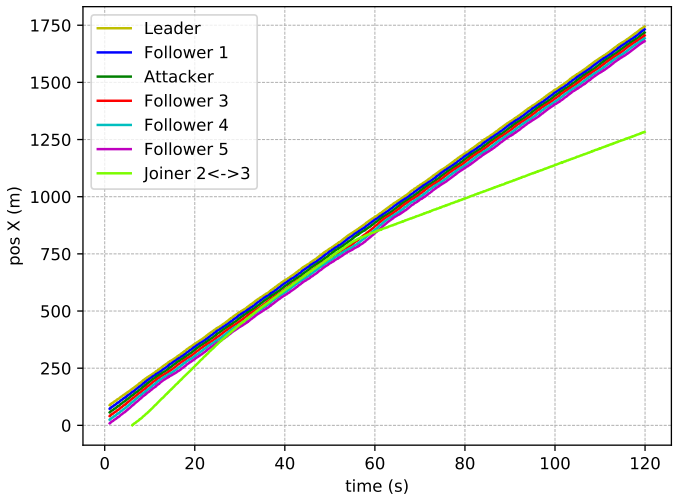
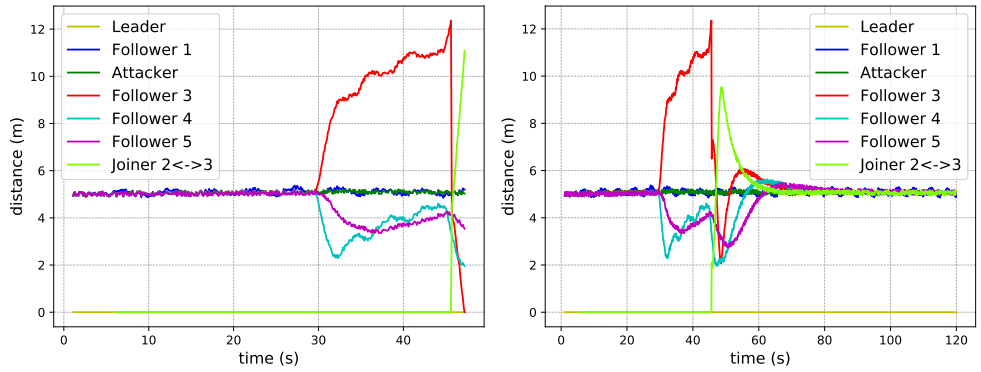


Figure 4.11: Join Denial of Entry for Ploeg During Gradual Speed Falsification.

of each vehicle. If we follow the green line (*Joiner*) and the red line (Follower 3/*JFollower*) we can see the result of the attack. Ploeg, as we saw earlier, is unaffected by this attack. Nonetheless, the maneuver is never completed because PATH (green) is forced to break due to the falsified value transmitted by the attacker. This causes him to undershoot the platoon which causes a maneuver abort. Similar results were produced from several attacks that affect the PATH controller.

Another result that we need to present, which is also related to the joining vehicle, is shown in Figure 4.12. The *Joiner* (green) approaches the platoon in order to enter between vehicle two and three. At the 45 seconds mark, the point where the *JFollower* is about to reach his required gap, the attacker (dark green) initiates the falsification. The *Joiner* is affected but he manages to enter the formation. In the negative falsification scenario, the vehicle is already decelerating by the attack and the Flatbed controller, which is not affected by acceleration values, is not fast enough to reverse it. We can see that the gap continues to close until the vehicles crash. During a positive acceleration the vehicle enters closer than the nominal distance but Flatbed manages to immediately reverse the forward momentum which causes some instability with his follower but ultimately avoids the collision. PATH had similar results but, as also shown here, it is susceptible to such an attack and thus the collisions can not be attributed to the same reason. Nonetheless, this is a disadvantage of the CVS policy which requires only small gaps between the vehicles.



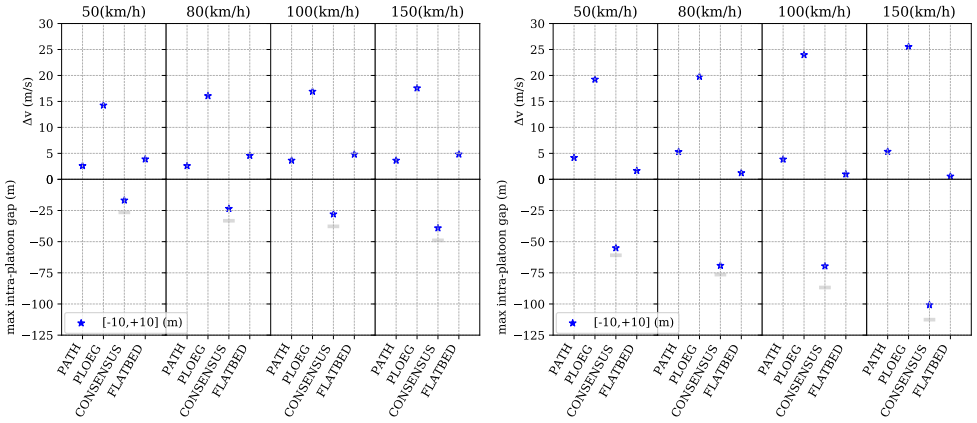
(a) Intra-platoon Gaps for Negative Acceleration

(b) Intra-platoon Gaps for Positive Acceleration

Figure 4.12: Acceleration Falsification on Flatbed During Join.

At this point, as with the exit maneuver, we delve into the effect our attacks had on the join procedure. During a smart position falsification the severity of the attacks is lower when it comes to both PATH and Flatbed; this small discrepancy appears either due to the *Joiner* entering closer to the *JFollower* or because the attack results in a forward collision which happens more gradually. Consensus manages to avoid any collision and is only destabilized, but the entry is denied due to the attack. Ploeg, on the other hand, performs worse as illustrated in Figure 4.13. The difference is around 20 to 30 km/h, giving a final ΔV of 70 to 95 km/h which can be fatal. This increase in potency happens because the *JFollower* has increased his distance from the attacker in order for the *Joiner* to enter, which aborts the entry due to the attack, leading to an increase in speed; hence the severity also increases.

The next two attacks we discuss, correspond to a simple acceleration falsification and the more sophisticated gradual acceleration attack. This way we can see the difference between static and gradually building values. In Figure 4.14 we detail the former. Consensus is unaffected completely while Flatbed crashes during the maneuver, as we already saw in Figure 4.12. PATH produces better results with the positive accelerations due to the increased gaps prepared for the entry. The entry never happens and the victim slams onto the attacker with greater speeds. The opposite is true for the negative acceleration. The maneuver finishes and the vehicles try to break, causing cars downstream to crash. In Ploeg's case, the process also finishes and the results are similar to the exit maneuver, albeit with less potency. For the positive values, because the crash is caused by the *Joiner* and not the *JFollower*, whereas for the negative



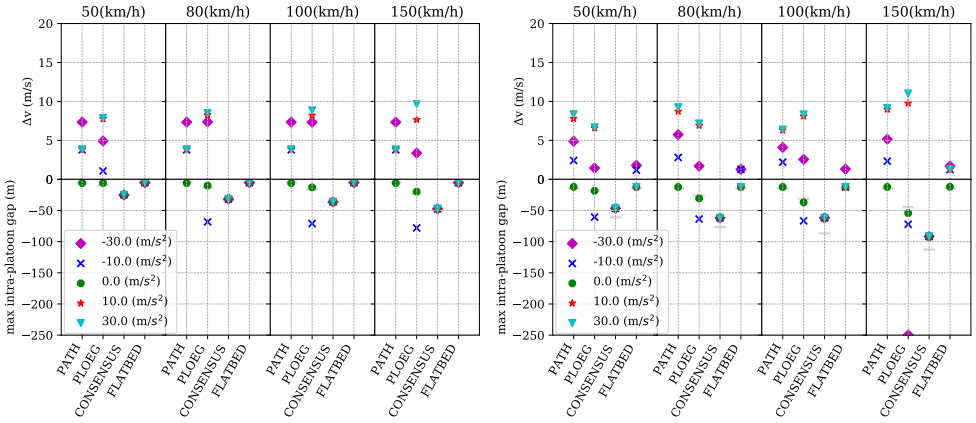
(a) Falsification during a Common Scenario (b) Falsification during a Join Maneuver

Figure 4.13: Smart Position Falsification on Join Maneuver.

ones, the breaking leads vehicle downstream to crash. These vehicles are one vehicle further downstream leading to slower breaking speeds.

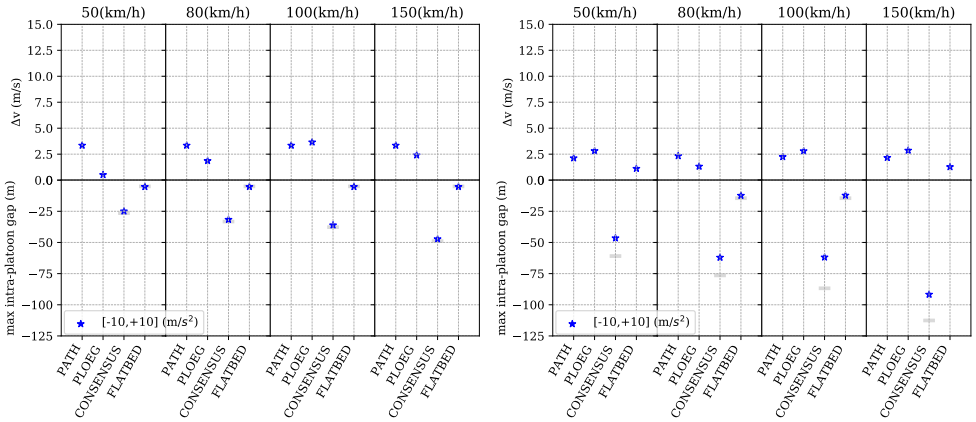
The first think we observe here, in Figure 4.15, is that the attacks are considerably less potent compared to the simple acceleration attack. The most severe attack causes a collision at 12 km/h which is small. Flatbed also appears to avoid the crashes at the medium speeds (80 and 100 km/h). In both cases, the cause is the same. The attack builds gradually and thus the vehicles change their speed in a slower pace, even when breaking. In comparison to the same attack during a normal platoon operation, the results are comparable; the only difference is Flatbed which is unaffected, as expected, by the acceleration attack.

The final plot for the join procedure compares the smart speed falsification during a common scenario and the join maneuver. The common scenario is detailed in Section 4.2.2 (Figures 4.4 and 4.5) and thus we jump straight into the new scenario. We can see that the smart attack loses steam in this test. PATH crashes a bit softer because the vehicle that crashes, same as in the common scenario, is already affected by the join procedure and has smaller space between him and his follower; the difference between the scenarios is only one meter. Flatbed also crashes, but only in the lowest speed. At this speed the controller is affected because the falsified value corresponds to large portion of the vehicles' current speed. The biggest difference observed in Figure 4.16 corresponds to the Ploeg controller. While in the common scenario, at the higher speeds, the severity increases, during the join maneuver the severity is only four times bigger (compared to twelve times) reaching a



(a) Falsification during a Common Scenario (b) Falsification during a Join Maneuver

Figure 4.14: Acceleration Falsification on Join Maneuver.



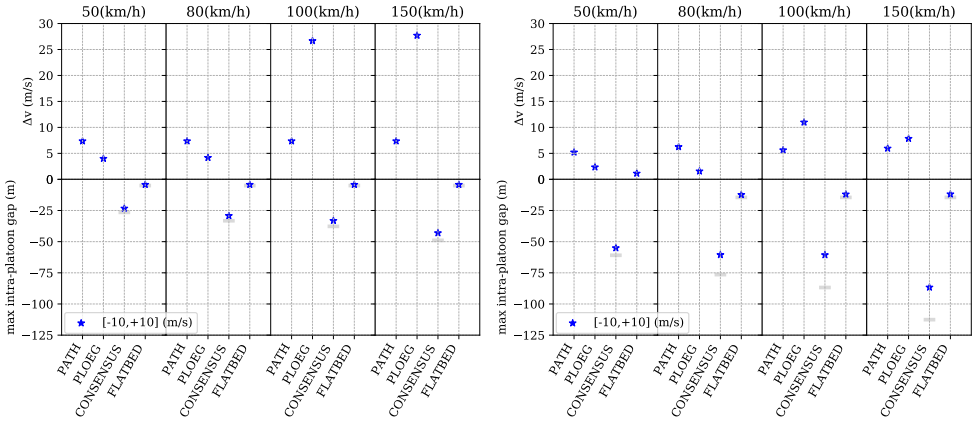
(a) Falsification during a Common Scenario (b) Falsification during a Join Maneuver

Figure 4.15: Gradual Acceleration Falsification During Join Maneuver.

ΔV of 40 km/h. This difference appears because of the controller change. At that moment, the acceleration flattens momentarily resulting in a maximum intra-platoon distance of 105 meters (compared to 125 meters) which leads to a lower final speed when the victim accelerates and collides.

4.2.5 Common Attacks

We have already presented some common attacks in order to compare our own results but here we showcase some attacks that we believe are interesting

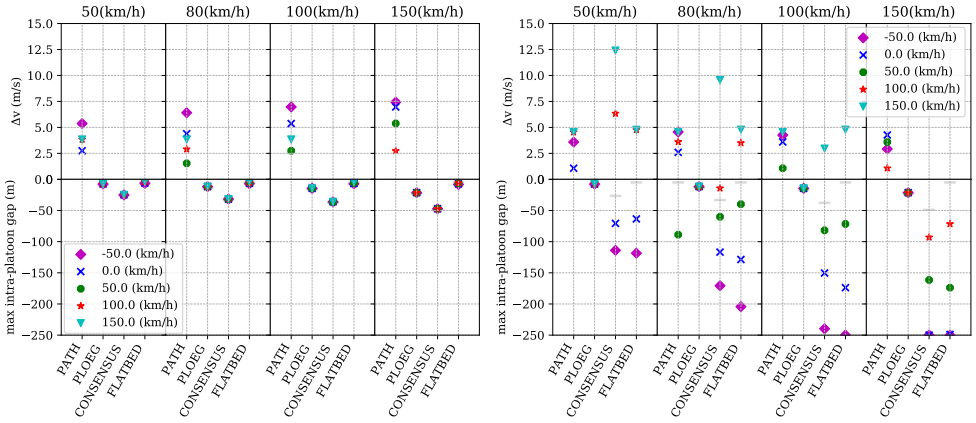


(a) Falsification during a Common Scenario (b) Falsification during a Join Maneuver

Figure 4.16: Smart Speed Falsification During Join Maneuver.

for the reader. As we have seen already, speed attacks are very potent. This time, we compare the speed injection originating from a follower, shown also in Figure 4.10, to the one from a malicious leader. In Figure 4.17.b we can see the value of having an attacker as the leader of the platoon. PATH results are similar, we a small loss in severity, because all the vehicles are attacked at the same time. This gives them more time to anticipate the breaking of their followers, leading to a crash that is less severe. Ploeg, is exceptionally resilient in speed attacks and remains unaffected here. Consensus and Flatbed are affected regardless of the speed and the falsified value with different results. We can see the difference that speed can make with the Consensus controller, the positive attacks loose steam as the platoon's speed increases because the distances increase and the relative difference between the attack value and the platoon speed becomes smaller. Conversely, the negative falsified values become more potent the faster the platoon is moving. The biggest intra-platoon gap reaches the extreme value of 250 meters. The situation is similar with Flatbed, with the difference that the positive values cause the same crash regardless of the moving pace of the formation.

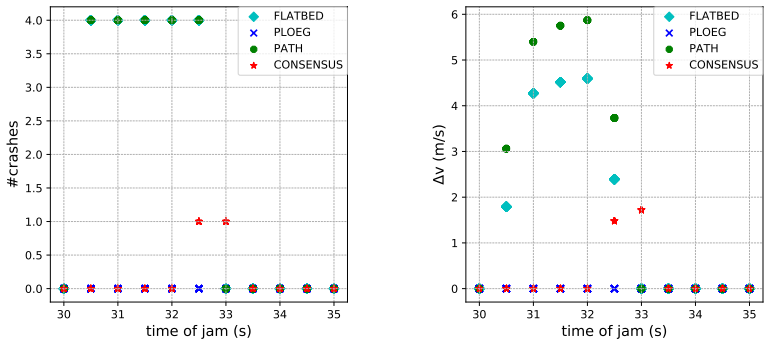
In Figure 4.18, we compare the effectiveness of a jamming attack when the attack is performed by the first vehicle. That is to say, no vehicle can gather information from the platoon leader. The time frame of the attack corresponds to acceleration for the range 30-32.5 seconds while from 32.5-35 the platoon is decelerating. In our work, we also include the Flatbed controller which has not been tested before in a jamming attack and the results are conclusive. PATH and Flatbed crash for every test that the platoon is increasing its speed.



(a) Follower Attacks (b) Leader Attacks

Figure 4.17: Speed Falsification Comparison.

In these scenarios, Ploeg only crashes sometimes while Consensus collides consistently when the acceleration starts to decrease. For both of them, the crashes happen at the lowest platoon speed. In every other case they mitigate the crash by increasing their intra-platoon gaps significantly. From the above we can conclude that speed does not affect the results and the most contributing factor is spacing, which Ploeg and Consensus compute based on the speed they are traveling. The latter two also appear to barely crash considering the low δV.



(a) Crashes based on Attack Start Times (b) Severity of Crashes

Figure 4.18: Controller Comparison Under Jamming.

4.3 Discussion

We can extrapolate several things from the results above, but it is easier to visualize a condensed summary; Tables 4.1 4.2 and 4.3, showcase the different effects our experiments have on each controller. We regard as instability any discrepancy between the nominal spacing and the perceived, even if that difference is relatively small. That discrepancy needs to continue throughout the attack, otherwise we deem the controller resilient. For example, during a join maneuver we do not regard the different intra-platoon distances from the vehicles downstream (from the *JFollower*) as instability. This deviation is needed as it is part of the controllers' response to the *JFollower's* movement.

PATH appears to perform poorly in most of the experiments. The intriguing result with PATH is that, either it crashes as a result of an attack or it is not affected at all; with only a tiny fractions resulting in instability as seen in Table 4.2. We can see that the other three are generally affected by a specific value, e.g. Ploeg is largely affected by acceleration attacks, but smart attacks can affect them even when the focus is not a value that they consider in their control algorithm. Furthermore, we can deduce two interesting results from the tables. Consensus appears to not be affected by a gradual position falsification; the reason for this is simple. The simple attacks continuously scale as time passes reaching in the end higher values than those disseminated by the more intelligent ones. The other remarkable observation is that the gradual attacks, in regards to controllers' resiliency, do not outperform the simpler attacks apart from a small potency in the acceleration falsification. Considering the potential countermeasures, though, the simple attacks can be safely ignored because the current defensive mechanisms (including the Kalman Filter) can detect them consistently; this is not necessarily true when a maneuver takes place. A countermeasure that takes into account all the kinematic properties, on the other hand, would make the gradual attacks obsolete. In that case, we can say that Consensus performs the best out of the four by avoiding acceleration crashes and mitigating most of the positional and speed falsifications.

But susceptibility to the attacks does not provide the best picture when we consider that the attacker may want to reuse his attacks. Some of them lead to the perpetrating vehicle crashing which may not be desirable; an attacker that is driving the vehicle or one who wants to reuse the same compromised vehicle at a later time (even when not present) would deem it not suitable. In Table 4.4 we gather all the attacks that cause a crash. Each cell notes the falsified value used, that led to the crash, and it is associated with a color. Green cells correspond to

Table 4.1: The percentage (%) of attacks that had no effect on the controllers.

Controller	Simple			Gradual			Combined		
	P	S	A	P	S	A	P	S	A
PATH	100	—	20	100	—	—	—	—	—
Ploeg	100	100	20	100	100	—	—	—	—
Consensus	—	75	100	—	75	100	—	—	20
Flatbed	100	—	68.75	100	—	62.5	—	—	75

Table 4.2: The percentage (%) of attacks that caused an instability.

Controller	Simple			Gradual			Combined		
	P	S	A	P	S	A	P	S	A
PATH	—	1.47	10	—	—	—	—	—	—
Ploeg	—	—	20	—	—	—	—	—	—
Consensus	—	19.11	—	43.75	—	—	75	87.5	75
Flatbed	—	77.95	20	—	75	25	—	68.75	—

Table 4.3: The percentage (%) of attacks that resulted in a crash.

Controller	Simple			Gradual			Combined		
	P	S	A	P	S	A	P	S	A
PATH	—	98.53	70	—	100	100	100	100	100
Ploeg	—	—	60	—	—	100	100	100	100
Consensus	100	5.88	—	56.25	25	—	25	12.5	—
Flatbed	—	22.05	11.25	—	25	12.5	100	31.25	25

collisions that do not involve the attacker himself and thus are preferable; when we have mixed results in terms of crashes, we prioritize for the best attacks from the point of view of the attacker (green cells). A red cell corresponds to a successful attacks that includes the misbehaving vehicle, but it also means that this is the best attack for this scenario (otherwise we would depict a green cell). The *no Maneuver* column has an extra detail compared to the other two, the values are preceded by the letters F or L; F stands for *Follower*, while L stands for *Leader*. Because it is possible for both cases to have a successful attack, we chose to present the most potent if they achieve different results, otherwise we include both to denote that the position of the attacker does not change the result. A single attack, on PATH, is accompanied by an asterisk (*) to denote a significant detail; we discuss this detail below. This hitmap provides any interested party a good starting point both in developing

misbehavior mechanisms and in designing new controllers.

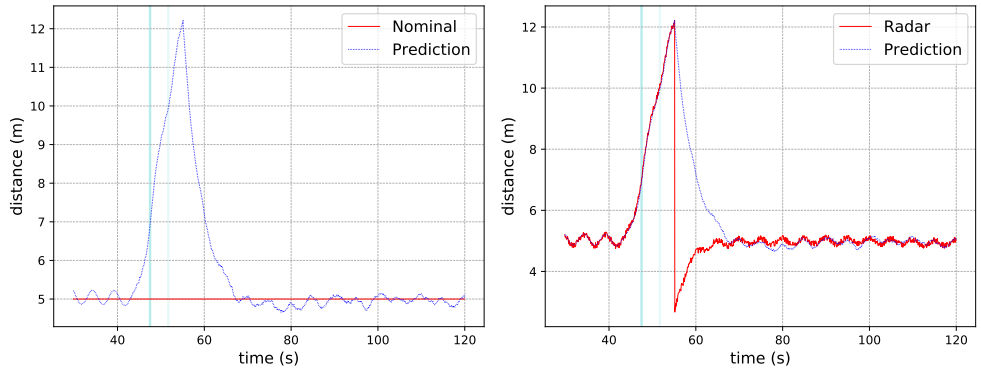
Table 4.4: Attacker's Potency Hitmap showing the crashes and prioritizing those that do not involve the attacker. Green: The attacker is not part of the collision; Red: The attacker is part of the crash; F and L denote a Follower or a Leader attacker; asterisk(*) denotes that there is a less potent attack that has some extra merit.

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}	{3,5,7,9,11}	{3,5,7,9,11}	{3,5,7,9,11}	{3,5,7,9,11}	{3,5,7,9,11}	{3,5,7,9,11}	{3,5,7,9,11}	{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}	{-50,0}	{-50,0,50}	{-50,0,50}	{-50,0,50,100}	{-50,0}	{-50,0,50}	{-50,0,50}	{-50,0,50,100}
	Consensus	L {100,150}	L 150	L 150		-	-	-	-	-	-	-	-
	Flatbed	L {100,150}	L {100,150}	L 150	-	{-50,0}	{-50,0}	{-50,0}	{-50,0,50,100}	-	-	-	-
	Ploeg	-	-	-	-	-	-	-	-	-	-	-	-
Acceleration (m/s^2)	PATH	F {-30,-10}	F {-30,-10}	F {-30, -10}	F {-30, -10}	{-30,-10}	{-30,-10}	{-30,-10}	{-30,-10}	{-30,-10}	{-30,-10}	{-30,-10}	{-30,-10}
	Consensus	-	-	-	-	-	-	-	-	-	-	-	-
	Flatbed	-	-	-	-	-	-	-	-	-	-	-	-
	Ploeg	LF {-10,-30}	LF -30	LF -30	LF -30	-30	-30	-30	{10,30}	-30	-30	-30	{10,30}
Gradual Position (m)	PATH	-	-	-	-	-	-	-	-	-	-	-	-
	Consensus	LF -10/+40	LF -10/+40	-	-	-10/+40	-10/+40	-	-	-10/+40	-10/+40	-	-
	Flatbed	-	-	-	-	-	-	-	-	-	-	-	-
	Ploeg	-	-	-	-	-	-	-	-	-	-	-	-
Gradual Speed (m/s)	PATH	LF {-10/+17}	LF {-10/+17}	LF {-10/+17}	LF {-10/+17}	-10/+17	-10/+17	-10/+17	-10/+17	-10/+17	-10/+17	-10/+17	-10/+17
	Consensus	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	-	-	-	-	-	-	-	-
	Ploeg	-	-	-	-	-	-	-	-	-	-	-	-
Gradual Acceleration (m/s^2)	PATH	F -10/+10	F -10/+10	F -10/+10	F -10/+10	-10/+10	-10/+10	-10/+10	-10/+10	-10/+10	-10/+10	-10/+10	-10/+10
	Consensus	-	-	-	-	-	-	-	-	-	-	-	-
	Flatbed	-	-	-	-	-10/+10	-	-	-10/+10	-	-	-	-
	Ploeg	LF -10/+10	LF -10/+10	LF -10/+10	LF -10/+10	-10/+10	-10/+10	-10/+10	-10/+10	-10/+10	-10/+10	-10/+10	-10/+10
Smart Position (m)	PATH	F* -10/+10	F* -10/+10	F* -10/+10	F* -10/+10	-10/+10	-10/+10	-10/+10	-10/+10	-10/+10	-10/+10	-10/+10	-10/+10
	Consensus	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	-10/+10	-10/+10	-10/+10	-10/+10	-10/+10	-10/+10	-10/+10	-10/+10
	Ploeg	L -10/+10	LF -10/+10	-10/+10	LF -10/+10	-10/+10	-10/+10	-10/+10	-10/+10	-10/+10	-10/+10	-10/+10	-10/+10
Smart Speed (m/s)	PATH	F -10/+10	F -10/+10	F -10/+10	F -10/+10	-10/+10	-10/+10	-10/+10	-10/+10	-10/+10	-10/+10	-10/+10	-10/+10
	Consensus	L -10/+10	L -10/+10	-	-	-	-	-	-	-	-	-	-
	Flatbed	L -10/+10	L -10/+10	L -10/+10	L -10/+10	-10/+10	-	-	-	-	-	-	-
	Ploeg	LF -10/+10	LF -10/+10	L -10/+10	L -10/+10	-10/+10	-10/+10	-10/+10	-10/+10	-10/+10	-10/+10	-10/+10	-10/+10
Smart Acceleration (m/s^2)	PATH	F -10/+10	F -10/+10	F -10/+10	F -10/+10	-10/+10	-10/+10	-10/+10	-10/+10	-10/+10	-10/+10	-10/+10	-10/+10
	Consensus	-	-	-	-	-	-	-	-	-	-	-	-
	Flatbed	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	-10/+10	-10/+10	-10/+10	-10/+10	-10/+10	-10/+10	-10/+10	-10/+10

Typically the attacks have the same results,regardless if a maneuver exist, with some exceptions. Consensus, for example, is collision free during a maneuver unless it involves a position falsification or a gradual position falsification for the lower speeds. PATH appears to be rather susceptible to attacks that involve either relatively negative speeds or negative accelerations. Those attacks, in all but one of the attacks, produce the best possible result for the attacker. For PATH, the smart position attack has an extra interesting detail. Although a followers attack is more potent, because the attacker is not involved, the results show that when the leader is initiating the attack a special case arises. More than one vehicles start to accelerate with the follower of the immediate victim also closing the gap fast. The victim crashes onto the attacker with a ΔV of 55 km/h (Figure 4.3.b) resulting in a worse attack, but his follower is also closely behind with a high speed. The resulting crash can be regarded as critical because of the potential "sandwich" situation that can emerge. Consensus proves to be the most difficult target for an attacker. All the attacks that lead to a crash, lead to one that involves the attacker (red cell). Also, for some attacks, like the gradual position and smart speed falsification, higher speeds do not cause an accident; the collisions are avoided thanks to the increased gaps between the vehicles. Flatbed, as we saw already, presents an interesting case. Certain attacks that do not yield good results in a normal scenario, produce potent attacks during the join procedure. Ploeg appears to

have mixed results in terms of potential attacks from an intelligent and self-preserving attacker. Some of the attacks that lead to a crash also fail to yield good results at highers speeds. This is attributed to the increased spacing between the vehicles; a clear advantage of the CTH policy over CVS. Finally, we can see that the position of the attacker is important when deploying his attacks. Surprisingly, being a follower allows for better attacks if the aim is to avoid a collision with the compromised car; the only attack that is solely valuable for a leader attacker is the smart position attack targeting Ploeg at the lowest speed.

One of our goals was also to test if the proposed, in the literature, kalman filter is adequate to in a platooning scenario. We already saw in Figure 2.5 (Section 2.4.3) the filter flag an attack when we initiated a speed falsification. Figure 4.19.a shows the internal detector of the *JFollower* that compares the nominal distance with the one expected by the kalman filter. The kalman filter is always updated with the previous state and so we see it moving upwards following the intra-platoon gap widening, which is required for the join procedure to continue. This is flagged at around the 47 second mark and a little later (51 second) the follower of the vehicle opening the gap also flags his predecessor (the *JFollower*) as a malicious node. In (b) we see the radar detector of the vehicles. The distance deviates from the expected values and the detector once again flags the vehicle. In this case, two vehicles of the platoon are marked as misbehaving nodes although the operation is benign. Similar results appear when we perform the exit maneuver. Consequently, the mechanism although promising is not, at this stage, adequate to be implemented in a platooning environment.

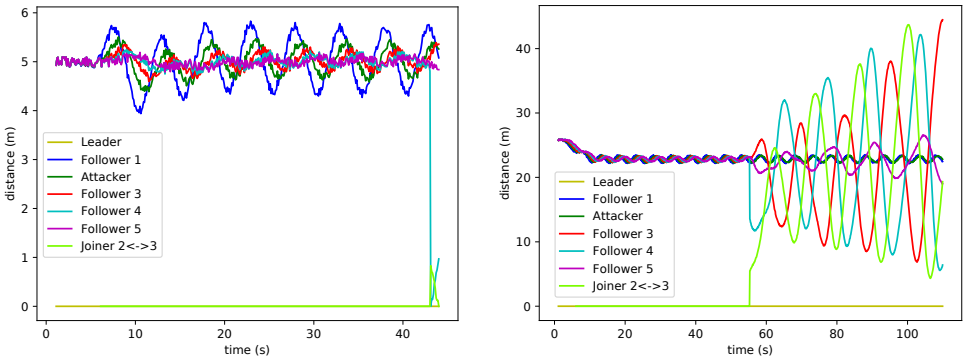


(a) Nominal Distance Detector (b) Radar Distance Detector

Figure 4.19: Kalman Filter During Join.

4.4 Observed Pitfalls

Before moving on to the conclusion of this work, we opt to highlight some interesting results that we came upon during the thesis. They may seem naive and wrong at first glance but it a scenario that can potentially happen on the road. It may be the case that an obstacle appears suddenly on the road ahead of the *Joiner*. The safest approach could be to change lane to avoid colliding with it; the only available lane is the lane of the platoon, subsequently, the *Joiner* tries to enter at any cost. Consider for example, that there is no obstacle ahead but a malicious actor performed a Sybil attack to confuse the vehicle. The end result is, string instability or worse, a collision between the vehicles. Figure 4.20 and Figure 4.21 illustrate what transpires when entering in a different position than the one specified; this is also a continuation of what we discussed briefly when describing the maneuver implementation in Section 3.2.2.



(a) PATH Join

(b) Ploeg Join

Figure 4.20: Join at Later Position.

In the former case, we can see a join maneuver performed by PATH and Ploeg respectively. The attacks are not relevant per se, the *Joiner* increases his intra-platoon gap because of an attack; the result is for him to enter between *Follower 3* and *Follower 4* which is not his assigned position. The later tries to slow down, to adhere to its spacing policy, while the *Joiner* tries to speed up, colliding with the vehicle which is supposed to create the gap. Ploeg, in comparison, is more resilient because of its increased spacing. That said, the vehicles try to increase their gaps, unsuccessfully, which leads to the deterioration of the platoon stability. Because this entry was not expected

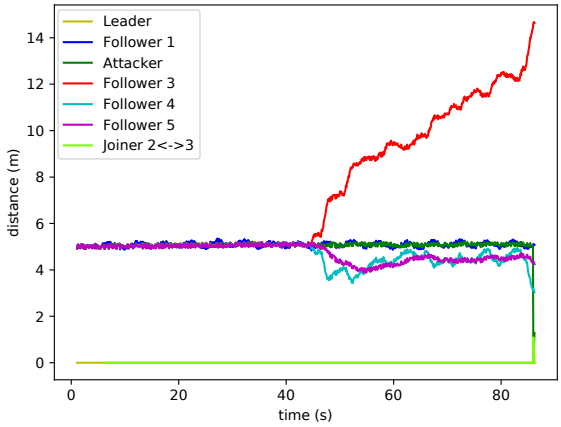


Figure 4.21: Join at Earlier Position for Flatbed.

by the vehicles, the protocol behaves as if the vehicles are in the correct position; thus the instability is amplified continuously between the *Joiner* and the followers. Figure 4.21, on the other hand, shows a join that happens after the *Joiner* overshoots its entry position because of an attack. In this case, he manages to enter ahead of the attacker; the gaps between the vehicles are so small, even for a CVS controller, which combined with the error in the perceived formation, lead to an immediate collision between the vehicles. Arguably, even if the protocol managed to compensate for the wrong position entry, the small irregularities in the sensor along with the extremely small gap would still lead to a collision.

Chapter 5

Conclusions and Future Work

5.1 Conclusions

What began as a plan to discover possible network attacks on vehicular platoons quickly formed into several research questions in regards to internal attacks in the scope of ITS. Attacks on the platoon application of these systems is separated into internal and external mainly because the width of the available ways to misbehave is vast. We already saw in Chapter 2 the standards put in place to safeguard the identity management and the privacy of the vehicles, but this can be expanded, both in credentials management and other areas, depending on the goal of the project. In our case, we aimed to find the limits of the different CACC controllers through extensive testing without constraining ourselves on the way the attacks can be performed.

By this point, we believe that we have reached several conclusions in that regard; speed falsifications can be very damaging for the platoons, with increasing spacing errors depending on the formation speed, knowledge that is valuable for any potential malicious actor. Attacks perpetrated by the platoon leader proved to be very potent when it comes to the collision's severity and this places concrete weight on the way platoon leaders are elected. We also show that in certain cases a follower attack is preferred if the malicious actor wants to preserve the compromised vehicle. That said, we show that a plethora of falsification attacks can be very damaging and at the same time favorable for the attacker. We also saw that despite the good performance of certain controllers like Flatbed and Consensus, none managed to avoid all the attacks. Consensus control algorithm is not infallible and an intelligent attacker can create problems whereas Flatbed is vulnerable at handling vehicle deceleration. Furthermore, due to the network topologies followed by the

controllers, the attacks can aim specific vehicles. This coupled with a join maneuver that allows entering in any place of the formation can pose a serious threat. Ploeg and PATH controllers were increasingly susceptible to this. That said, relying only on a particular case of joining is limiting because the platoon leader can elect to only allow entering from the back, thus, diminishing the potential implications; unless the attacker has no regard about the vehicle from which he performs the injection attack, in which case several attacks can still succeed. That said, the closer to the leader a vehicle is, the better its fuel consumption is. Furthermore, platoon homogeneity which is promoted by the middle join is also preferred, thus it is prudent to not hastily reach a conclusion on the matter.

The only limiting factor in our work is the fact that we simulate the experiments instead of performing them on actual roads with real vehicles. We tried to simulate sensor inaccuracies and jittering in the platoon but it is possible that a real world scenario could induce more uncertainties. Despite this, we are confident that out work met the goals it set at the beginning and provides a thorough investigation on the matter at hand.

5.2 Future Work

We managed to introduce several attacks that were missing from the literature and we reached new insights with our experiments; but further work can be done on the matter. This can include the combinations of different type of attacks, potential countermeasures to mitigate them or experiments with different vehicle properties.

We initiated our attacks at a moment where the platoon was accelerating because it can provide better results for the attacker, but we have not touched upon attacks that can start when the platoon is decelerating. This distinction is important because certain injections used positive values which can potentially lead to different results when the following vehicles are not increasing their speed; we believe this can increase the instability of the platoon. Similarly, our gradual attacks started with negative values first and progressively increased. The opposite, after what we have seen, is probably more devastating; an accelerating vehicle responds worse to fluctuating kinematic properties. In the same manner, jamming attacks have been the focus of many experiments in the literature, we also performed some of them, but using them in conjunction with our attacks has not been done before. For example, both Ploeg and Consensus are mostly unaffected by jamming but it stills incurs a spacing error on the platoon which an attacker can use in relation to a falsification attack to provoke

an accident. Several of our attacks, also, induced instability to the controllers by making them close the intra-platoon gaps; in cases likes these, targeted jamming attacks that only interfere with one or two beacons could potentially be the tipping stone in inducing a collision. We believe, Flatbed and Consensus controllers to be the relevant controllers in such a case.

We performed our experiments by gradually changing the kinematic values during a maneuver, starting with negative values. That said, a more thorough investigation can be done in attacks that mimic the space creation needed for a join, thus, masking the attack from defensive mechanisms such as kalman filters; even in cases where the model based misbehavior mechanism is suited to comprehend a maneuver. This has the potential of colliding different vehicles together without the involvement of the attacker. Moreover, our maneuver scenarios include an attacker that resides in the platoon but is not the leader of the formation. This can provide some interesting results considering the susceptibility that some controllers have to the leader's falsified data. Flatbed, for example, which is prone to invalid speed data originating from the leader is a prime target for such an experiment. A step further, would be to have more than one malevolent vehicles that try to attack the platoon in a coordinated manner. This could lead to some very potent attacks especially when we consider the maneuvers presented here; the leader could coordinate with the special follower, the one who makes space, to provoke an accident aimed towards the joiner. The same applies to the rather ineffective exit maneuver attacks. Collaborating vehicles can disseminate opposite values to destabilize the platoon and lead vehicles to crash with higher ΔVs than the ones shown here.

With our diverse framework for initiating falsification attacks, a followup to this work is the evaluation of misbehavior detection mechanisms that exist in the literature. In Chapter 2 we briefly touched upon vehicle localization but the idea can be expanded to assist the relevant detection mechanisms in deducing the trustworthiness of the data transmitted by the vehicles; thus, giving the control algorithms an extra tool to discern erroneous data. In their work, the RSUs help in creating proofs of location, at a rate of 10 Hz, to discern Sybil nodes; in our case we can use the same principles to better evaluate if the reported values are correct by detecting discrepancies in the actual location of the vehicles based on the stored data. This is crucial because such an approach does not require a kalman filter, which are ineffective during maneuvers, although, a combination of all these mechanisms can be very advantageous towards misbehavior detection; each mechanism can supplement the other in detecting an attack.

Finally, our tests were made with uniform platoons in terms of length and engine characteristics, but this is not always the case; especially with the increasing numbers of electrical vehicles. Thus, we can include not only vehicles of different sizes but also cars that have variable engine properties which is a contributing factor in the responsiveness of the controllers.

References

[1] A. A. Alam, A. Gattami, and K. H. Johansson, "An experimental study on the fuel reduction potential of heavy duty vehicle platooning," in *International IEEE Conference on Intelligent Transportation Systems*, Sep. 2010. doi: 10.1109/ITSC.2010.5625054. ISSN 2153-0009 pp. 306–311. [Online]. Available: https://ieeexplore.ieee.org/document/5625054

[2] P. Varaiya, "Smart cars on smart roads: problems of control," *IEEE Transactions on Automatic Control*, vol. 38, no. 2, pp. 195–207, Feb 1993. doi: 10.1109/9.250509. [Online]. Available: https://ieeexplore.ieee.org/document/250509

[3] A. Taeihagh and H. S. M. Lim, "Governing autonomous vehicles: emerging responses for safety, liability, privacy, cybersecurity, and industry risks," *Transport Reviews*, vol. 39, no. 1, pp. 103–128, July 2019. doi: 10.1080/01441647.2018.1494640. [Online]. Available: https://doi.org/10.1080/01441647.2018.1494640

[4] E. C. T. Robinson, E. Chan, "Operating platoons on public motorways: an introduction to the sartre platooning programme," in *World Congress on Intelligent Transport Systems*, October 2010, pp. 1–11. [Online]. Available: https://www.webcitation.org/5vpgbawbW?url=http://www.sartre-project.eu/en/publications/Documents/SARTRE_Overview_Final_Paper_ITS_World_Congress_2010.pdf

[5] M. EL-Zaher, B. Dafflon, F. Gechter, and J.-M. Contet, "Vehicle platoon control with multi-configuration ability," *Procedia Computer Science*, vol. 9, pp. 1503 – 1512, June 2012. doi: https://doi.org/10.1016/j.procs.2012.04.165 Proceedings of the International Conference on Computational Science, ICCS 2012. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1877050912002864

[6] T. ETSI, "Intelligent transport systems (its); vehicular communications; basic set of applications; definitions," Tech. Rep., 6 2009. [Online]. Available: https://www.etsi.org/deliver/etsi_tr/102600_102699/102638/01.01.01_60/tr_102638v010101p.pdf

[7] J. Axelsson, T. Bergh, A. Johansson, B. Mårdberg, P. Svenson, and V. Åkesson, "Truck platooning business case analysis," July 2020. [Online]. Available: http://www.diva-portal.org/smash/record.jsf?pid=diva2%3A1410842&dswid=2238

[8] A. Festag, "Cooperative intelligent transport systems standards in europe," *Communications Magazine, IEEE*, vol. 52, pp. 166–172, 12 2014. doi: 10.1109/MCOM.2014.6979970. [Online]. Available: https://ieeexplore.ieee.org/document/6979970

[9] P. Papadimitratos, A. D. La Fortelle, K. Evenssen, R. Brignolo, and S. Cosenza, "Vehicular communication systems: Enabling technologies, applications, and future outlook on intelligent transportation," *IEEE Communications Magazine*, vol. 47, no. 11, pp. 84–95, November 2009. doi: 10.1109/MCOM.2009.5307471. [Online]. Available: https://ieeexplore.ieee.org/document/5307471

[10] "Ieee standard for wireless access in vehicular environments–security services for applications and management messages," *IEEE Std 1609.2-2016 (Revision of IEEE Std 1609.2-2013)*, pp. 1–884, March 2016. doi: 10.1109/IEEESTD.2016.7426684. [Online]. Available: https://ieeexplore.ieee.org/document/7544433

[11] W. Whyte, A. Weimerskirch, V. Kumar, and T. Hehn, "A security credential management system for v2v communications," 12 2013. doi: 10.1109/VNC.2013.6737583. ISBN 978-1-4799-2687-9 pp. 1–8. [Online]. Available: https://www.researchgate.net/publication/271554151_A_security_credential_management_system_for_V2V_communications

[12] M. Khodaei and P. Papadimitratos, "The key to intelligent transportation: Identity and credential management in vehicular communication systems," *IEEE Vehicular Technology Magazine*, vol. 10, no. 4, pp. 63–69, December 2015. doi: 10.1109/MVT.2015.2479367. [Online]. Available: https://ieeexplore.ieee.org/document/7317862

[13] M. Khodaei, H. Jin, and P. Papadimitratos, "Towards deploying a scalable robust vehicular identity and credential management infrastructure," in *IEEE Vehicular Networking Conference (VNC)*, December 2014. doi: 10.1109/VNC.2014.7013306 pp. 33–40. [Online]. Available: https://ieeexplore.ieee.org/document/7013306

[14] M. Khodaei, H. Jin, and P. Papadimitratos, "Secmace: Scalable and robust identity and credential management infrastructure in vehicular communication systems," *IEEE Transactions on Intelligent Transportation Systems*, vol. 19, no. 5, pp. 1430–1444, April 2018. doi: 10.1109/TITS.2017.2722688. [Online]. Available: https://ieeexplore.ieee.org/document/8332521

[15] M. Khodaei, H. Noroozi, and P. Papadimitratos, "Scaling Pseudonymous Authentication for Large Mobile Systems," in *Proceedings of the ACM conference on Security and privacy in wireless & mobile networks (WiSec)*, Miami, FL, USA, May 2019. doi: https://doi.org/10.1145/3317549.3323410 pp. 174–184. [Online]. Available: https://dl.acm.org/doi/10.1145/3317549.3323410

[16] "Car-to-car communication consortium (c2c-cc)," February 2011. [Online]. Available: https://www.car-2-car.org/

[17] M. S. Al-kahtani, "Survey on security attacks in vehicular ad hoc networks (vanets)," in *International Conference on Signal Processing and Communication Systems*, December 2012. doi: 10.1109/ICSPCS.2012.6507953 pp. 1–9. [Online]. Available: https://www.researchgate.net/publication/260739919_Survey_on_security_attacks_in_Vehicular_Ad_hoc_Networks_VANETs

[18] R. Mishra, A. Singh, and R. Kumar, "Vanet security: Issues, challenges and solutions," in *International Conference on Electrical, Electronics, and Optimization Techniques (ICEEOT)*, March 2016. doi: 10.1109/ICEEOT.2016.7754846. ISSN null pp. 1050–1055. [Online]. Available: https://ieeexplore.ieee.org/document/7754846

[19] S. M. Safi, A. Movaghar, and M. Mohammadizadeh, "A novel approach for avoiding wormhole attacks in vanet," in *Second International Workshop on Computer Science and Engineering*, vol. 2, October 2009. doi: 10.1109/WCSE.2009.787 pp. 160–165. [Online]. Available: https://ieeexplore.ieee.org/document/5403263

[20] ISO, "Road vehicles – Functional safety," 11 2011.

[21] Z. Zhong and J. Lee, "The effectiveness of managed lane strategies for the near-term deployment of cooperative adaptive cruise control," *ArXiv*, vol. abs/1908.10404, November 2019. doi: https://doi.org/10.1016/j.tra.2019.08.015. [Online]. Available: https://www.sciencedirect.com/science/article/abs/pii/S0965856419303520

[22] M. Amoozadeh, H. Deng, C.-N. Chuah, H. M. Zhang, and D. Ghosal, "Platoon management with cooperative adaptive cruise control enabled by vanet," *Vehicular Communications*, vol. 2, no. 2, pp. 110 – 123, April 2015. doi: https://doi.org/10.1016/j.vehcom.2015.03.004. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S2214209615000145

[23] R. Hall and C. Chin, "Vehicle Sorting for Platoon Formation: Impacts on Highway Entry and Throughput," *Transportation Research Part C: Emerging Technologies*, vol. 13, no. 5, pp. 405–420, October 2005. doi: https://doi.org/10.1016/j.trc.2004.09.001. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0968090X06000040

[24] T. Dao, C. M. Clark, and J. P. Huissoon, "Optimized Lane Assignment Using Inter-Vehicle Communication," in *IEEE Intelligent Vehicles Symposium*, June 2007. doi: 10.1109/IVS.2007.4290284 pp. 1217–1222. [Online]. Available: https://ieeexplore.ieee.org/document/4290284

[25] A. Mihály and P. Gaspar, "Control of Platoons Containing Diverse Vehicles with the Consideration of Delays and Disturbances," *Periodica Polytechnica Transportation Engineering*, vol. 40, pp. 21–26, January 2012. doi: 10.3311/pp.tr.2012-1.04. [Online]. Available: https://www.researchgate.net/publication/275565682_Control_of_platoons_containing_diverse_vehicles_with_the_consideration_of_delays_and_disturbances

[26] H. C. Joksch, "Velocity change and fatality risk in a crash–a rule of thumb," *Accident Analysis & Prevention*, vol. 25, no. HS-042 059, February 1993. doi: 10.1016/0001-4575(93)90102-3. [Online]. Available: https://pubmed.ncbi.nlm.nih.gov/8420529/

[27] B. Wiedersheim, Z. Ma, F. Kargl, and P. Papadimitratos, "Privacy in inter-vehicular networks: Why simple pseudonym change is

not enough," in *Seventh International Conference on Wireless On-demand Network Systems and Services (WONS)*, February 2010. doi: 10.1109/WONS.2010.5437115 pp. 176–183. [Online]. Available: https://ieeexplore.ieee.org/document/5437115

[28] L. Buttyán, T. Holczer, A. Weimerskirch, and W. Whyte, "Slow: A practical pseudonym changing scheme for location privacy in vanets," in *IEEE Vehicular Networking Conference (VNC)*, October 2009. doi: 10.1109/VNC.2009.5416380 pp. 1–8. [Online]. Available: https://ieeexplore.ieee.org/document/5416380

[29] J. Freudiger, M. Raya, M. Félegyházi, P. Papadimitratos, and J.-P. Hubaux, "Mix-zones for location privacy in vehicular networks," Januray 2007. [Online]. Available: https://www.researchgate.net/publication/37450059_Mix-Zones_for_Location_Privacy_in_Vehicular_Networks

[30] C. Vaas, M. Khodaei, P. Papadimitratos, and I. Martinovic, "Nowhere to hide? mix-zones for private pseudonym change using chaff vehicles," in *IEEE Vehicular Networking Conference (VNC)*, December 2018. doi: 10.1109/VNC.2018.8628449 pp. 1–8. [Online]. Available: https://ieeexplore.ieee.org/document/8628449

[31] M. Khodaei and P. Papadimitratos, "Cooperative location privacy in vehicular networks: Why simple mix-zones are not enough," *IEEE Internet Of Things Journal*, 2021.

[32] J. R. Douceur, "The sybil attack," in *Peer-to-Peer Systems*, P. Druschel, F. Kaashoek, and A. Rowstron, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, October 2002. doi: https://doi.org/10.1007/3-540-45748-8_24. ISBN 978-3-540-45748-0 pp. 251–260. [Online]. Available: https://link.springer.com/chapter/10.1007/3-540-45748-8_24

[33] M. Khodaei and P. Papadimitratos, "Evaluating on-demand pseudonym acquisition policies in vehicular communication systems," in *Proceedings of the First International Workshop on Internet of Vehicles and Vehicles of Internet*, ser. IoV-VoI '16. New York, NY, USA: Association for Computing Machinery, 2016. doi: 10.1145/2938681.2938684. ISBN 9781450343459 p. 7–12. [Online]. Available: https://doi.org/10.1145/2938681.2938684

[34] F. Schaub, F. Kargl, Z. Ma, and M. Weber, "V-tokens for conditional pseudonymity in vanets," in *IEEE Wireless*

Communication and Networking Conference, April 2010. doi: 10.1109/WCNC.2010.5506126 pp. 1–6. [Online]. Available: https://ieeexplore.ieee.org/document/5506126

[35] P. Papadimitratos, L. Buttyan, J. Hubaux, F. Kargl, A. Kung, and M. Raya, "Architecture for secure and private vehicular communications," in *International Conference on ITS Telecommunications*, June 2007. doi: 10.1109/ITST.2007.4295890 pp. 1–6. [Online]. Available: https://ieeexplore.ieee.org/document/4295890

[36] J. J. Haas, Y. Hu, and K. P. Laberteaux, "Efficient certificate revocation list organization and distribution," *IEEE Journal on Selected Areas in Communications*, vol. 29, no. 3, pp. 595–604, February 2011. doi: 10.1109/JSAC.2011.110309. [Online]. Available: https://ieeexplore.ieee.org/document/5719271

[37] A. Uchikawa, R. Hatori, T. Kuroki, and H. Shigeno, "Filter multicast: A dynamic platooning management method," in *IEEE Consumer Communications and Networking Conference*, Jan 2010. doi: 10.1109/CCNC.2010.5421617. ISSN 2331-9860 pp. 1–5. [Online]. Available: https://ieeexplore.ieee.org/document/5421617?section=abstract

[38] M. Khodaei and P. Papadimitratos, "Efficient, Scalable, and Resilient Vehicle-Centric Certificate Revocation List Distribution in VANETs," in *Proceedings of the ACM conference on Security and privacy in wireless & mobile networks*, Stockholm, Sweden, June 2018. doi: 10.1145/3212480.3212481 pp. 172–183. [Online]. Available: https://www.diva-portal.org/smash/get/diva2:1231581/FULLTEXT01.pdf

[39] ——, "Scalable & Resilient Vehicle-Centric Certificate Revocation List Distribution in Vehicular Communication Systems," *IEEE Transactions on Mobile Computing (IEEE TMC)*, pp. –, March 2020. doi: 10.1109/TMC.2020.2981887. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/9042314

[40] H. Jin and P. Papadimitratos, "Dos-resilient cooperative beacon verification for vehicular communication systems," *Ad Hoc Networks*, vol. 90, p. 101775, July 2019. doi: https://doi.org/10.1016/j.adhoc.2018.10.003 Recent advances on security and privacy in Intelligent Transportation Systems.

[Online]. Available: http://www.sciencedirect.com/science/article/pii/S1570870518307108

[41] M. Amoozadeh, A. Raghuramu, C. Chuah, D. Ghosal, H. M. Zhang, J. Rowe, and K. Levitt, "Security vulnerabilities of connected vehicle streams and their impact on cooperative driving," *IEEE Communications Magazine*, vol. 53, no. 6, pp. 126–132, June 2015. doi: 10.1109/MCOM.2015.7120028. [Online]. Available: https://ieeexplore.ieee.org/document/7120028

[42] B. Rosenberg, *Handbook of financial cryptography and security*. CRC Press, August 2010. [Online]. Available: http://docshare02.docshare.tips/files/26397/263973091.pdf

[43] Z. El-Rewini, K. Sadatsharan, D. F. Selvaraj, S. J. Plathottam, and P. Ranganathan, "Cybersecurity challenges in vehicular communications," *Vehicular Communications*, vol. 23, p. 100214, June 2020. doi: https://doi.org/10.1016/j.vehcom.2019.100214. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S221420961930261X

[44] B. Ribeiro, F. Gonçalves, A. Santos, M. J. Nicolau, B. Dias, J. Macedo, and A. Costa, "Simulation and testing of a platooning management protocol implementation," in *Wired/Wireless Internet Communications*, Y. Koucheryavy, L. Mamatas, I. Matta, A. Ometov, and P. Papadimitriou, Eds. Cham: Springer International Publishing, June 2017. doi: https://doi.org/10.1007/978-3-319-61382-6_14. ISBN 978-3-319-61382-6 pp. 174–185. [Online]. Available: https://www.researchgate.net/publication/318131337_Simulation_and_Testing_of_a_Platooning_Management_Protocol_Implementation

[45] ETSI-TR-103-298, "Intelligent Transport Systems (ITS); Platooning; Pre-standardisation Study," https://portal.etsi.org/webapp/WorkProgram/Report_WorkItem.asp?wki_id=44191, ETSI, Tech. Rep., January 2019.

[46] B. Gerrits, M. Mes, and P. Schuur, "Simulation of real-time and opportunistic truck platooning at the port of rotterdam," in *Proceedings of the Winter Simulation Conference*, ser. WSC '19. IEEE Press, December 2019. doi: 10.1109/WSC40007.2019.9004852. ISBN 9781728132839 p. 133–144. [Online]. Available: https://ieeexplore.ieee.org/document/9004852

[47] B. Zarrouki, V. Klös, M. Grabowski, and S. Glesner, in *Fault-Tolerance by Graceful Degradation for Car Platoons*, 03 2019. doi: 10.4230/OASIcs.ASD.2019.1. [Online]. Available: https://www.researchgate.net/publication/332072423_Fault-Tolerance_by_Graceful_Degradation_for_Car_Platoons

[48] Y. Zheng, S. Eben Li, J. Wang, D. Cao, and K. Li, "Stability and scalability of homogeneous vehicular platoon: Study on the influence of information flow topologies," *IEEE Transactions on Intelligent Transportation Systems*, vol. 17, no. 1, pp. 14–26, Jan 2016. doi: 10.1109/TITS.2015.2402153. [Online]. Available: https://ieeexplore.ieee.org/document/7055887

[49] R. W. Thomas and J. M. Vidal, "Ad hoc vehicle platoon formation," in *SoutheastCon*, April 2019. doi: 10.1109/SoutheastCon42311.2019.9020370 pp. 1–5. [Online]. Available: https://ieeexplore.ieee.org/document/9020370

[50] E. Nunen, D. Tzempetzis, G. Koudijs, H. Nijmeijer, and M. Brand, "Towards a safety mechanism for platooning," 06 2016. doi: 10.1109/IVS.2016.7535433 pp. 502–507. [Online]. Available: https://ieeexplore.ieee.org/document/7535433

[51] M. Saad, M. Abdel-Aty, J. Lee, and L. Wang, "Access design safety analysis for managed lanes including accessibility level and weaving length," November 2018. doi: 10.1061/JTEPBS.0000191. [Online]. Available: https://www.researchgate.net/publication/327527240_Safety_Analysis_of_Access_Zone_Design_for_Managed_Toll_Lanes_on_Freeways

[52] Q. Cai, M. Saad, M. Abdel-Aty, J. Yuan, and J. Lee, "Safety impact of weaving distance on freeway facilities with managed lanes using both microscopic traffic and driving simulations," *Transportation Research Record*, vol. 2672, no. 39, pp. 130–141, June 2018. doi: 10.1177/0361198118780884. [Online]. Available: https://doi.org/10.1177/0361198118780884

[53] M. Asplund, "Poster: Securing vehicular platoon membership," in *IEEE Vehicular Networking Conference (VNC)*, December 2014. doi: 10.1109/VNC.2014.7013324 pp. 119–120. [Online]. Available: https://ieeexplore.ieee.org/document/7013324

[54] S. U. Hussain and F. Koushanfar, "Privacy preserving localization for smart automotive systems," in *ACM/EDAC/IEEE Design Automation Conference (DAC)*, June 2016. doi: 10.1145/2897937.2898071 pp. 1–6. [Online]. Available: https://dl.acm.org/doi/10.1145/2897937.2898071

[55] F. Boeira, M. Asplund, and M. Barcellos, "Vouch: A secure proof-of-location scheme for vanets," 10 2018. doi: 10.1145/3242102.3242125 pp. 241–248. [Online]. Available: https://www.ida.liu.se/labs/rtslab/publications/2018/Felipe_MSWiM.pdf

[56] D. Swaroop and K. R. Rajagopal, "A review of constant time headway policy for automatic vehicle following," in *IEEE Intelligent Transportation Systems*, August 2001. doi: 10.1109/ITSC.2001.948631 pp. 65–69. [Online]. Available: https://ieeexplore.ieee.org/document/948631

[57] A. Ghiasi, O. Hussain, Z. S. Qian, and X. Li, "A mixed traffic capacity analysis and lane management model for connected automated vehicles: A Markov chain method," *Transportation Research Part B: Methodological*, vol. 106, no. C, pp. 266–292, December 2017. doi: 10.1016/j.trb.2017.09.022. [Online]. Available: https://ideas.repec.org/a/eee/transb/v106y2017icp266-292.html

[58] D. Swaroop and J. K. Hedrick, "Constant Spacing Strategies for Platooning in Automated Highway Systems," *Journal of Dynamic Systems, Measurement, and Control*, vol. 121, no. 3, pp. 462–470, 09 1999. doi: 10.1115/1.2802497. [Online]. Available: https://doi.org/10.1115/1.2802497

[59] R. Rajamani, *Vehicle dynamics and control*. Springer Science & Business Media, December 2011. [Online]. Available: https://www.springer.com/gp/book/9781461414322

[60] S. Santini, A. Salvi, A. S. Valente, A. Pescapé, M. Segata, and R. Lo Cigno, "A consensus-based approach for platooning with intervehicular communications and its validation in realistic scenarios," *IEEE Transactions on Vehicular Technology*, vol. 66, no. 3, pp. 1985–1999, June 2016. doi: 10.1109/TVT.2016.2585018. [Online]. Available: https://ieeexplore.ieee.org/document/7499858

[61] A. Ali, G. Garcia, and P. Martinet, "The flatbed platoon towing model for safe and dense platooning on highways," *IEEE Intelligent*

Transportation Systems Magazine, vol. 7, no. 1, pp. 58–68, January 2015. doi: 10.1109/MITS.2014.2328670. [Online]. Available: https://ieeexplore.ieee.org/document/7014426

[62] J. Ploeg, B. T. M. Scheepers, E. van Nunen, N. van de Wouw, and H. Nijmeijer, "Design and experimental evaluation of cooperative adaptive cruise control," in *International IEEE Conference on Intelligent Transportation Systems (ITSC)*, October 2011. doi: 10.1109/ITSC.2011.6082981 pp. 260–265. [Online]. Available: http://www.dct.tue.nl/New/Wouw/ITSC2011_Ploeg.pdf

[63] H. Hao and P. Barooah, "Stability and robustness of large platoons of vehicles with double-integrator models and nearest neighbor interaction," *International Journal of Robust and Nonlinear Control*, vol. 23, no. 18, pp. 2097–2122, December 2013. doi: 10.1002/rnc.2872. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1002/rnc.2872

[64] A. Alipour-Fanid, M. Dabaghchian, H. Zhang, and K. Zeng, "String stability analysis of cooperative adaptive cruise control under jamming attacks," in *IEEE International Symposium on High Assurance Systems Engineering (HASE)*, January 2017. doi: 10.1109/HASE.2017.39 pp. 157–162. [Online]. Available: https://ieeexplore.ieee.org/document/7911891

[65] R. van der Heijden, T. Lukaseder, and F. Kargl, "Analyzing attacks on cooperative adaptive cruise control (cacc)," in *IEEE Vehicular Networking Conference (VNC)*, November 2017. doi: 10.1109/VNC.2017.8275598 pp. 45–52. [Online]. Available: https://ieeexplore.ieee.org/document/8275598

[66] M. Iorio, F. Risso, R. Sisto, A. Buttiglieri, and M. Reineri, "Detecting injection attacks on cooperative adaptive cruise control," in *IEEE Vehicular Networking Conference (VNC)*, December 2019. doi: 10.1109/VNC48660.2019.9062798 pp. 1–8. [Online]. Available: https://ieeexplore.ieee.org/document/9062798

[67] R. W. van der Heijden, S. Dietzel, T. Leinmüller, and F. Kargl, "Survey on misbehavior detection in cooperative intelligent transportation systems," *IEEE Communications Surveys Tutorials*, vol. 21, no. 1, pp. 779–811, October 2018. doi: 10.1109/COMST.2018.2873088. [Online]. Available: https://ieeexplore.ieee.org/document/8477005

[68] S. Marti, T. J. Giuli, K. Lai, and M. Baker, "Mitigating routing misbehavior in mobile ad hoc networks," in *Proceedings of the Annual International Conference on Mobile Computing and Networking*, ser. MobiCom '00. New York, NY, USA: Association for Computing Machinery, August 2000. doi: 10.1145/345910.345955. ISBN 1581131976 p. 255–265. [Online]. Available: https://doi.org/10.1145/345910.345955

[69] R. van der Heijden, T. Lukaseder, and F. Kargl, "Veremi: A dataset for comparable evaluation of misbehavior detection in vanets," 04 2018. doi: https://doi.org/10.1007/978-3-030-01701-9_18. [Online]. Available: https://arxiv.org/abs/1804.06701

[70] K. Garlichs, A. Willecke, M. Wegner, and L. C. Wolf, "Trip: Misbehavior detection for dynamic platoons using trust," in *IEEE Intelligent Transportation Systems Conference (ITSC)*, October 2019. doi: 10.1109/ITSC.2019.8917188 pp. 455–460. [Online]. Available: https://ieeexplore.ieee.org/document/8917188

[71] V. Milanés, S. E. Shladover, J. Spring, C. Nowakowski, H. Kawazoe, and M. Nakamura, "Cooperative adaptive cruise control in real traffic situations," *IEEE Transactions on Intelligent Transportation Systems*, vol. 15, no. 1, pp. 296–305, February 2014. doi: 10.1109/TITS.2013.2278494. [Online]. Available: https://ieeexplore.ieee.org/document/6588305

[72] X.-Y. Lu and S. E. Shladover, *Automated Truck Platoon Control and Field Test*. Cham: Springer International Publishing, August 2014, pp. 247–261. ISBN 978-3-319-05990-7. [Online]. Available: https://doi.org/10.1007/978-3-319-05990-7_21

[73] M. Segata, S. Joerer, B. Bloessl, C. Sommer, F. Dressler, and R. Lo Cigno, "PLEXE: A Platooning Extension for Veins," in *IEEE Vehicular Networking Conference (VNC 2014)*. Paderborn, Germany: IEEE, 12 2014. doi: 10.1109/VNC.2014.7013309. ISBN 978-1-4799-7660-7. ISSN 2157-9857 pp. 53–60. [Online]. Available: https://www.researchgate.net/publication/269108890_Plexe_A_platooning_extension_for_Veins

[74] P. A. Lopez, M. Behrisch, L. Bieker-Walz, J. Erdmann, Y. Flötteröd, R. Hilbrich, L. Lücken, J. Rummel, P. Wagner, and E. Wiessner,

"Microscopic traffic simulation using sumo," in *International Conference on Intelligent Transportation Systems (ITSC)*. IEEE, November 2018. doi: 10.1109/ITSC.2018.8569938 pp. 2575–2582. [Online]. Available: https://ieeexplore.ieee.org/document/8569938

[75] C. Sommer, R. German, and F. Dressler, "Bidirectionally Coupled Network and Road Traffic Simulation for Improved IVC Analysis," *IEEE Transactions on Mobile Computing (TMC)*, vol. 10, no. 1, pp. 3–15, January 2011. doi: 10.1109/TMC.2010.133. [Online]. Available: https://ieeexplore.ieee.org/document/5510240

[76] A. Varga and R. Hornig, "An overview of the omnet++ simulation environment," 01 2008. doi: 10.1145/1416222.1416290 p. 60. [Online]. Available: https://dl.acm.org/doi/10.5555/1416222.1416290

Appendix A

Maneuvers Implementation

The maneuvers' protocol are explained in Section 2.2.1; here, the control flow and the message communication are detailed. In the figures depicting the protocols, for brevity we regard the message communications as Message (MSG) and the step-by-step process as Control Flow (CF).

A.1 Join Maneuver

Control Flow #1: Request to Join the platoon:

1. Check if we are already in a maneuver
2. Change our role to *JOINER* and our operational mode to *inManeuver*
3. Encapsulate {myId, platoonId, platoonLeaderId, myLaneNumber, myPosition, entryPosition} into message #1
4. Send message #1 to the platoon leader
5. Change inner state to *J_WAIT_REPLY*

Control Flow #2: The leader handles the request in the following manner:

1. Check if a join is allowed at this time
2. Send message #2: {myId, platoonId, requestVehicleId, join_allowed} to the requesting vehicle
3. Set our operational mode to *inManeuver*

4. Deactivate lane changes
5. Save {myLaneNumber, vehicleId, entryPosition} received from the request
6. Create a *new_formation* structure containing all the IDs including the vehicleId from the request in the appropriate position
7. Create and send message #3 to the requesting vehicle: {myId, platoonId, joinerId, platoonSpeed, platoonLane, new_formation, CACC_spacing, platoonController}
8. Change inner state to *L_WAIT_JOINER_IN_POSITION*

Control Flow #3: The joining vehicle handles the join request response in the following way:

1. Checks if the inner state corresponds to *J_WAIT_REPLY*
2. If Join maneuver is permitted, deactivate lane changes and change inner state to *J_WAIT_INFORMATION*; otherwise abort by resetting the role to *NONE*, changing the state to *IDLE* and deactivating the *inManeuver* operational mode

Control Flow #4: The joining vehicle handles the information message with these steps:

1. Check if our inner state is *J_WAIT_INFORMATION*
2. Save the data transmitted from the leader
3. Change our lane to that of the platoon if we enter from the back
4. Change to *FAKE_CACC* and set the spacing or headway depending on the controller; speedup to catch the platoon
5. Flip our inner state to *J_MOVE_IN_POSITION*

Control Flow #5: The vehicle approaches the platoon through these steps:

1. Check if our state is *J_MOVE_IN_POSITION* or *J_WAIT_JOIN*

2. Capture the beacon messages disseminated from the front of our position
3. Approach until our distance is the one required by the controller
4. If our state is *J_MOVE_IN_POSITION*
    (a) Encapsulate the following: {myId, platoonId, platoonLeaderId, platoonSpeed, platoonLane, new_formation} and send it to the platoon leader as message #4
    (b) Flip our inner state to *J_WAIT_JOIN*

Control Flow #6: The leader responds to the joiner being in correct position:

1. Check if inner state corresponds to *L_WAIT_JOINER_IN_POSITION*
2. Validate the new formation structure send by the entering vehicle
3. If the entry position is at the back
    (a) Encapsulate {myId, platoonId, joinerId, platoonSpeed, platoonLane, new_formation} into message #5 and send it to the joiner
4. If the entry position is in the middle
    (a) Encapsulate {myId, platoonId, jFollowerId, platoonSpeed, platoonLane} into message #6 and send it to the vehicle occupying the entry position
5. Change our inner state to *L_WAIT_JOINER_TO_JOIN*

Control Flow #7: The follower in the entry position handles the request to make space for the joiner:

1. Change the role to *JFOLLOWER*
2. Change the spacing values depending on the controller to make space

Control Flow #8: The jFollower opens the gap:

1. Capture the beacon messages disseminated from the front of our position

2. Increase the gap until our distance is the one required by the controller
3. Create message #7: {myId, platoonId, platoonLeaderId, platoonSpeed, platoonLane} and send it to the leader

Control Flow #9: The leader notifies the joiner:

1. Retrieve the jFollwer data
2. Create message #8: {myId, platoonId, joinerId, platoonSpeed, platoonLane new_formation} and send it to the vehilcle that wants to enter the platoon

Control Flow #10: The entering vehicle moves into the formation:

1. Check that we are in state *J_WAIT_JOIN*
2. Validate the formation sent by the leader
3. Change our lane if we are entering in the middle
4. Change our controller to the one supplied by the leader
5. Set our new formation inside the platoon
6. Create message #9: {myId, platoonId, platoonLeaderId, platoonSpeed, platoonLane, new_formation} and send it to the leader
7. Change our role to *FOLLOWER*, our state to *IDLE* and deactivate *inManeuver* mode

Control Flow #11: The leader finishes the maneuver with the below steps:

1. Check we are at state *L_WAIT_JOINER_TO_JOIN*
2. Validate incoming formation structure
3. If the vehicle entered in the middle
   (a) Notify the jFollower of the completion with message #10: {myId, platoonId, jFollowerId, platoonSpeed, platoonLane}
4. Set the new formation structure locally

5. Create an update beacon, message #11: {myId, platoonId, platoonSpeed, platoonLane, new_formation}
6. Disseminate message #11 to all the vehicles of the platoon
7. Change inner state to *IDLE* and deactivate *inManeuver* mode

Control Flow #12: The jFollower reverts to normal operation with the following:

1. Revert the spacing values in order for the vehicle to close the gap
2. Change the role to *FOLLOWER*

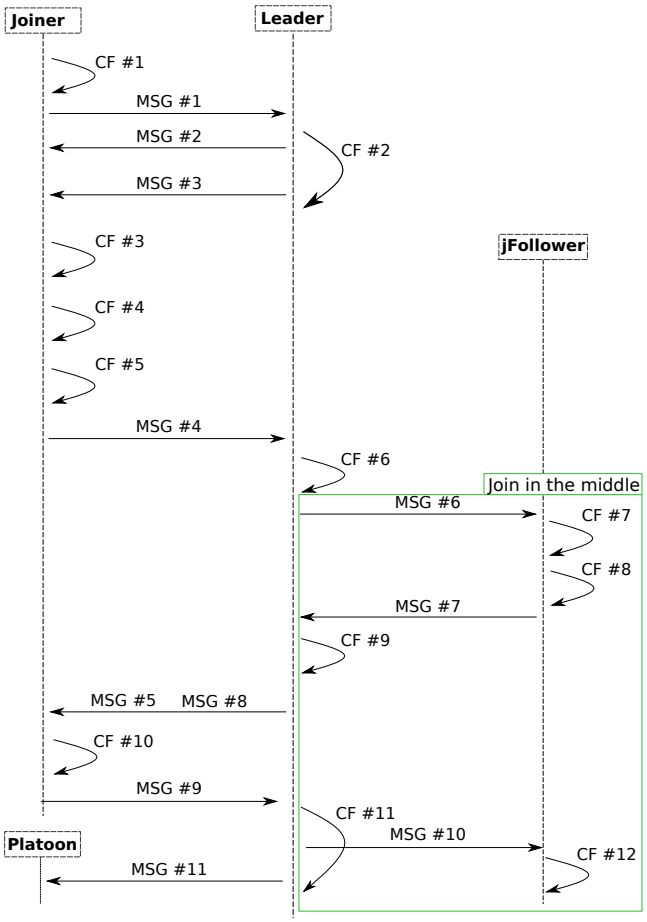


Figure A.1: UML Diagram of Join Maneuver.

A.2 Exit Maneuver

Control Flow #1: Request to Exit the platoon:

1. Check if we are already in a maneuver
2. Change our role to *LEAVER* and our operational mode to *inManeuver*
3. Encapsulate {myId, platoonId, platoonLeaderId, myLaneNumber, myPosition} into message #1
4. Send message #1 to the platoon leader
5. Change inner state to *E_WAIT_REPLY*

Control Flow #2: The leader handles the request in the following manner:

1. Check if an exit is allowed at this time
2. Send message #2: {myId, platoonId, requestVehicleId, exit_allowed} to the requesting vehicle
3. Set our operational mode to *inManeuver*
4. Deactivate lane changes
5. Save {myLaneNumber, vehicleId} received from the request
6. Create a *new_formation* structure containing all the IDs without the vehicleId who requested the Exit maneuver
7. Create and send message #3 to the requesting vehicle: {myId, platoonId, leaverId, platoonSpeed, platoonLane, new_formation}
8. Change inner state to *L_WAIT_LEAVER_IN_POSITION*

Control Flow #3: The exiting vehicle handles the exit request response in the following way:

1. Checks if the inner state corresponds to *E_WAIT_REPLY*
2. If Exit maneuver is permitted, deactivate lane changes and change inner state to *E_WAIT_INFORMATION*; otherwise abort by resetting the role to *FOLLOWER*, changing the state to *IDLE* and deactivating the *inManeuver* operational mode

Control Flow #4: The exiting vehicle handles the information message with these steps:

1. Check if our inner state is *E_WAIT_INFORMATION*
2. Save the data transmitted from the leader
3. Change our lane to the one next to the platoonLane
4. Change to *FAKE_CACC* until we leave the platoon
5. Flip our inner state to *E_MOVE_IN_POSITION*

Control Flow #5: Handle change lane:

1. Check if the state is *E_MOVE_IN_POSITION*
2. Capture the beacon messages disseminated from the front of our position
3. Check if the lane is changed by measuring the distances with our sensors
4. Encapsulate the following values: {myId, platoonId, platoonLeaderId, platoonSpeed, platoonLane, new_formation} and send it to the platoon leader as message #4
5. Flip our inner state to *E_WAIT_EXIT*

Control Flow #6: The leader instructs the vehicle to leave the platoon:

1. Check if inner state corresponds to *L_WAIT_LEAVER_IN_POSITION*
2. Validate the new formation structure send by the leaving vehicle
3. Encapsulate {myId, platoonId, leaverId, platoonSpeed, platoonLane, new_formation} into message #5 and send it to the leaver
4. Change our inner state to *L_WAIT_LEAVEVR_TO_EXIT*

Control Flow #7: The leaving vehices exits the platoon for good performing the following:

1. Check if inner state is *E_WAIT_EXIT*

2. Validate incoming formation with the one already stored
3. Change our controller to *ACC*
4. Update the local platoon information from the received values
5. Notify the leader with message #6 containing these: {myId, platoonId, platoonLeaderId, platoonSpeed, myLaneNumber, new_formation}
6. Finish the procedure by resetting the inner state to *IDLE*, the role to *NONE* and deactivate the operational mode *inManeuver*

Control Flow #8: The leader finishes the maneuver with the below steps:

1. Check we are at state *L_WAIT_LEAVEVR_TO_EXIT*
2. Validate incoming formation structure
3. Set the new formation structure locally
4. Create an update beacon, message #7: {myId, platoonId, platoonSpeed, platoonLane, new_formation}
5. Disseminate message #7 to all the vehicles of the platoon
6. Change inner state to *IDLE* and deactivate *inManeuver* mode

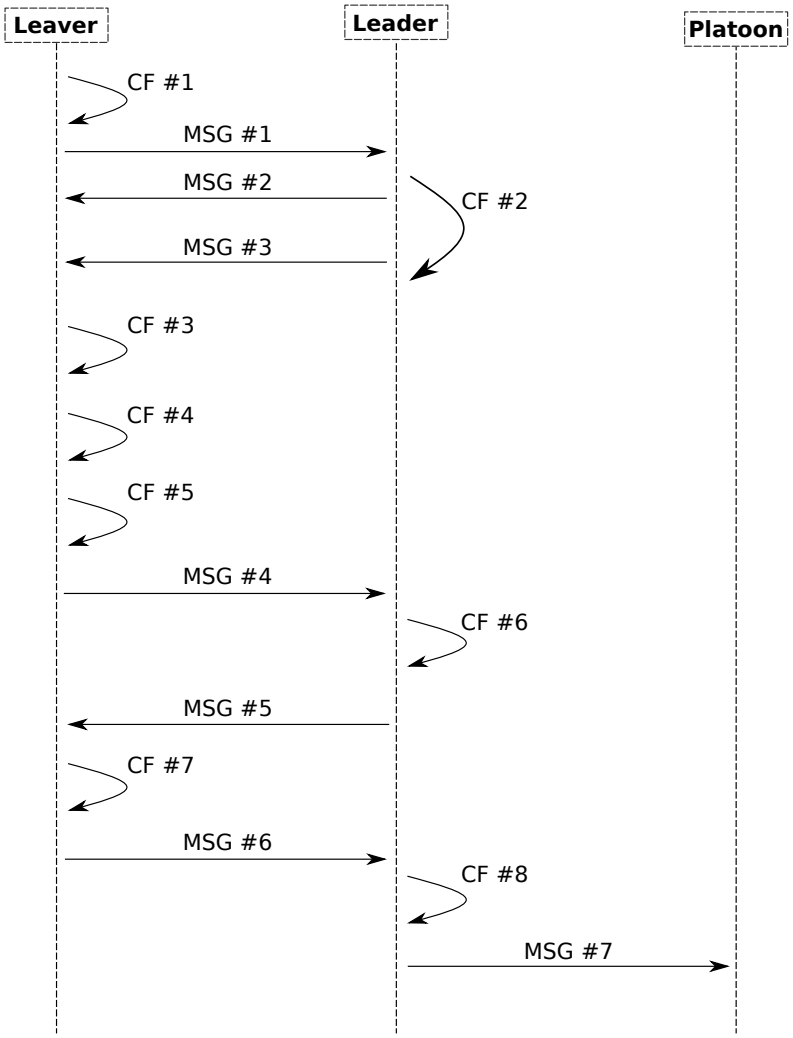


Figure A.2: UML Diagram of Exit Maneuver.

Appendix B

Configuration Parameters

Here we present a more detailed view of the configuration file used to perform our experiments. The configuration is separated into a global scope and a configuration scope, i.e., lines appearing after *[Config Maneuver]* are only applicable when this specific configuration is run by the program or when another configuration that extends it is, as seen in line 45. The modular way of our implementation is observed in several lines. Lines 9–11 indicate the use of filters for this application and the existence or not of realistic sensors while lines 33–42 showcase the way someone can configure the relevant positions used in the maneuvers and which, if at all, to perform. Lines 70–83 provide the means with which to choose what attack to perform and the values to experiment with.

```
1
2  ###########################################################
3  # Application #
4  ###########################################################
5
6  *.node[*].appl_type = "DetectionApp"
7
8  # use the kalman filter mechanism or not
9  *.node[*].appl.detector = "no"
10 #*.node[*].appl.detector = "kalmanfilter"
11 *.node[*].appl.realisticSensors = ${realisticSensors = true}
12
13 # detection thresholds
14 *.node[*].appl.fallbackACCHeadway = -1s
15 # *.node[*].appl.fallbackACCHeadway = 1.5s
16 *.node[*].appl.fallbackACCSafetyMargin = 0.5
17 *.node[*].appl.detectionAvgWindow = 10
18 *.node[*].appl.detectionAttackTolerance = 10
19 *.node[*].appl.distanceKFThresholdFactor = 0.33
20 *.node[*].appl.distanceRadarThresholdFactor = 0.25
21 *.node[*].appl.detectionAccelerationFactor = 0.05
22 *.node[*].appl.qFactor = 50
23
24 #enable statistics recording for the application
25 *.node[*].appl.*.scalar-recording = true
26 *.node[*].appl.*.vector-recording = true
27
28 [Config Maneuver]
29 # combine maneuvers with attacks
30 *.node[*].scenario_type = "CombinedScenario"
31
32 #use the join maneuver scenario
```

```
*.node[*].appl.joinManeuver = "no"
#*.node[*].appl.joinManeuver = "JoinAtPosition"
*.node[*].scenario.carPositionEnter = 3
#Perform maneuver
**.traffic.joinManeuver = false

#use the exit maneuver scenario
*.node[*].appl.exitManeuver = "no"
#*.node[*].appl.exitManeuver = "ExitAtPosition"
*.node[*].scenario.carPositionExit = 3

[Config InjectionAttack]
extends = RealisticSensors,Maneuver

repeat = 1
# set leader speeds and the controllers to test
*.node[*].scenario.caccSpacing = ${CACCSpacing = 5, 5, 5, 5 ! controller }
*.node[*].scenario.leaderSpeed = ${leaderSpeed = 50, 80, 100, 150}kmph
**.numericController = ${controller = 1, 2, 3, 4}
*.node[*].scenario.controller = ${sController = "CACC", "PLOEG", "CONSENSUS", "
   FLATBED" ! controller}

# set the time at which an attack needs to be initiated
*.node[*].appl.attackStart = ${attackStart = 40}s
*.node[*].prot.attackStart = ${attackStart}s

# which vehicle performs the attack
*.node[*].prot.attackingNode = ${attackingNode = 2}

**.traffic.platoonInsertDistance = ${5, 5, 5, 5 ! controller}m
# default values taken from SUMO
**.traffic.platoonInsertHeadway = ${0, 0.5, 0.8, 0 ! controller}s
**.traffic.platoonLeaderHeadway = ${leaderHeadway}s

[Config SinusoidalPosInjectionAttack]
extends = InjectionAttack

# choose an attack type here
*.node[*].appl.attackerType = ${attacker = 2}
*.node[*].prot.attackerType = ${attacker}

# the attack type is used to choose one of those options
*.node[*].prot.maliciousSpeed = ${attackSpeed = 150}kmph
*.node[*].prot.shiftX = ${attackPos = 3, 5, 7, 9, 11} m
*.node[*].prot.maliciousAcc = ${attackAcc = -30 } #mpss #strange fix
*.node[*].prot.fakeSpeed = ${fakeSpeed = ($attacker == 3)}
*.node[*].prot.fakeAcc = ${fakeAcc = ($attacker == 4)}
*.node[*].prot.fakePos = ${fakePos = ($attacker == 2)}
*.node[*].prot.gradualChange = ${gradualChange = ($attacker == 5)}
*.node[*].prot.smartAttack = ${smartAttack = ($attacker == 6)}
*.node[*].prot.lowLimit = -10
*.node[*].prot.upLimit = 10
```

Listing B.1: The configuration details and parameters used to conduct our tests.