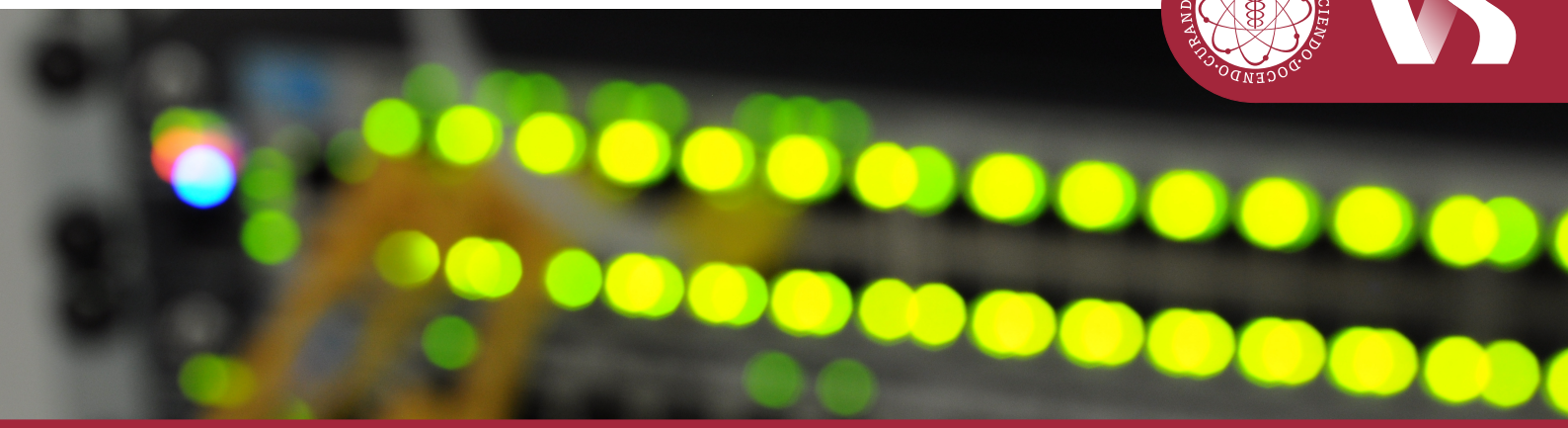


Data-Centric Misbehavior Detection in Platooning

Claus Ruepp
Bachelor's Thesis

Examiner: Prof. Dr. rer. nat. Frank Kargl
Supervisor: Dr. Rens van der Heijden
VS Number: VS-2019-B24
Submission Date: June 27, 2019

Issued: June 27, 2019

I hereby declare that this thesis titled:

Data-Centric Misbehavior Detection in Platooning

is the product of my own independent work and that I have used no sources or materials other than those specified. The passages taken from other works, either verbatim or paraphrased in the spirit of the original quote, are identified in each individual case by indicating the source.
I further declare that all my academic work was written in line with the principles of proper academic research according to the official “Satzung der Universität Ulm zur Sicherung guter wissenschaftlicher Praxis” (University Statute for the Safeguarding of Proper Academic Practice).

Ulm, June 27, 2019

Claus Ruepp, student number 905635

Abstract

Cooperative Adaptive Cruise Control (CACC) is a promising technology to help manage the ever increasing traffic volumes on the streets, and make automobile transportation more safe and efficient. While very promising, CACC faces many security risks, which are especially important because more often than not, the health and safety of passengers is at stake. Misbehavior Detection through semantic data filtering aims to minimize the risks of traffic participants being endangered in environments where CACC is actively being negatively influenced by proactive prevention of attack effects. Through simulations in Plexe, we try to show that data injection attacks can be recognised by the Maat detection framework, and negative effects can be mitigated simply by disregarding malicious messages. Using our simulations, we were not able to find improvements that can be attributed to the misbehavior detection. Despite our results, semantic misbehavior detection has enough potential room for improvements to be a great approach at making road transportations more secure in the future. We estimate that further work on the detection methods and parameters is likely to make the technology invaluable to keeping roads in a more autonomous future safe.

Contents

1 Introduction

Modern society relies on efficient transportation, but growing traffic volumes bring along additional complexity and susceptibility to congestions. Intelligent Transportation Systems (ITS) appear promising with regards to keeping transportation as efficient as possible, but at the same time require reliable and secure communication between traffic participants. Since a big share of data exchanged in such an ITS has high locality, opting for a rigid networking scheme with authoritarian hosts would only introduce unnecessary delays for message deliveries as well as a lot of additional strain on existing communications infrastructure, or even require huge investments into roadside infrastructure all around the world. Instead, Vehicular Ad-Hoc Networks (VANETs) are highly dynamic, decentralized networks of adjacent vehicles (Vehicle-to-Vehicle, V2V) and roadside equipment (Vehicle-to-Infrastructure, V2I), which satisfy these requirements. Typically, VANETs are formed as ad-hoc networks of vehicular On-Board Units (OBUs) and Road-Side Units (RSUs) [22]. RSUs are interconnected among each other as well as often connected to an Internet Service Providers (ISP). OBUs on the same ad-hoc network communicate with each other so as to exchange traffic or even other information, and can also connect to the internet via RSUs. V2V communication can happen across multiple hops in the same network, allowing vehicles to stay connected despite being physically out of range for a direct single-hop connection. Because traditional communication protocols are not suitable for networks with such volatile topologies, standards like IEEE 802.11p [11] have been devised. Since central authoritative systems are not necessarily reachable, and every client is potentially an intruder, ensuring security in VANETs proves to be a very difficult task. Such a VANET may act as the backbone to Automated Driving applications of different automation levels like Cooperative Adaptive Cruise Control (CACC) and Highly Automated Driving (HAD), for which vehicles may want to exchange information gathered through sensory systems or strategical instructions among each other in order to gain a better model of the current real world context, or work together to improve ride quality, traffic flow, and consequently, fuel efficiency. Due to their relevance for applications like CACC and HAD, where the physical or mental health of people may be negatively influenced, properly securing such VANETs against malicious interference is especially critical.

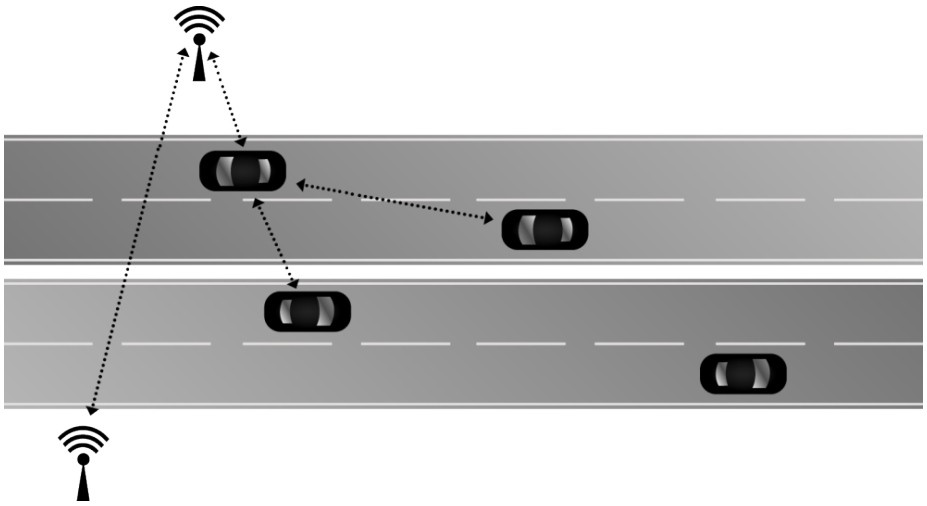


Figure 1.1: Example VANET topology

1.1 Motivation

Traditional public key infrastructure approaches are suited to prevent previous offenders and illegitimate entities from participating in V2V communication. Efficient management and revocation of certificates is one of the main challenges a PKI faces in the context of VANETs, where a central infrastructure may not always be reachable [26]. Attacks from clients within such networks (insiders) are still a possibility, and need to be taken care of. CACC applications require a combination of security mechanisms to be well equipped and able to protect traffic participants. Semantic data filters are an implementation of data-centric misbehavior detection, coupled with a proactive mitigation strategy, which has not been surveyed exhaustively yet, despite promising potential to detect data injection attacks and prevent harm to vehicle platoons. Negative effects due to undetected attacks require a second, reactive layer of protection, in the form of a controller module for example to mitigate the attack impact and return a platoon to a safe state. In this work, we intend to survey how semantic filters as a proactive security strategy can be integrated into CACC simulations, and whether or not there is a measurable improvement in ride quality with such filters in place.

1.2 Structure

In Chapter 2 we review threats to CACC applications, previously analysed attacks and different mitigation strategies. Chapter 3 shows the system- and attacker models we devised to exhibit the efficacy of semantic data filters with regards to misbehavior detection. We consider attacker models from previous work, and introduce the experimental set-up, and how we built it in chapter 4. In Chapter 5 results of our work will be

shown, analyzed and compared to previous findings. Implications and shortcomings of the work will be discussed in Chapter 6, and Chapter 7 will conclude the work, and give some thoughts for future work.

2 Related Work

Automated Driving applications rely heavily on communication between participating vehicles, and VANETs are a type of network which is well suited for such applications. These VANETs are heavily modular, decentralized networks between vehicles and occasionally infrastructure, and allow for local exchange of, for example, positional data [6]. Since CACC not only concerns capital investments (cars, goods) but more importantly also directly affects the health and safety of individuals, it needs to be secured well. In order to build a secure application, the underlying technology needs to be secure in the first place. VANETs behave somewhat like traditional computer networks with added volatility, and are consequentially exposed to similar threats [24] [4].

CACC applications on the other hand are susceptible to some additional threats besides jamming or other types of denial-of-service attacks. Simple message tampering attacks, where a malicious node sends beacons with altered velocity, position or acceleration information to other vehicles in a platoon, can destabilize the platoon and often even cause a crash [10] [3]. Executing such a tampering attack using one or multiple false identities (so-called Sybil nodes) to inject the falsified information further increases the effectiveness of such an attack while also making it harder to detect in time [2]. There are various approaches to detecting such misbehavior, but no single effort to detect attacks appears to be sufficient on their own. Based on the significant metric, misbehavior detection efforts can be divided into two groups - trust based approaches, which score each participating node in the platoon based on their adherence to protocols or data integrity, while another group focuses solely on the data and dismisses or approves beacons based on data plausibility or probability. [7] provides an overview over notable studies and their positive and negative aspects.

As [10] suggests, a combination of strong misbehavior detection and resilient control algorithms with active mitigation capabilities may be required to guarantee secure platoons. Conveniently, this work also provides attacker models, which we can expand upon and compare to. Petrillo et al. [19] propose and survey a control strategy with good mitigation potential, acting as a complimentary ”reactive” strategy to misbehavior detection. Noei [16] exhibits another successful control strategy which increases inter-vehicle distances if transmitted speeds of adjacent vehicles deviate from speeds gathered through sensory systems. In [17], an effective, dynamically updated threshold for returning control to the driver is presented, which could cooperate with other misbehavior detection mechanisms to leave an autonomous controller in control for longer, but still return control to the driver in situations that a controller may not be suited for, therefore further increasing ride comfort and driver downtime. Erickson explores a different

kind of misbehavior detection in [5]. He proposes contracts for autonomous vehicles, in which agreements are made on, for example, a maximum threshold for braking, and shows that such contracts can be quickly and safely dissolved in case of an emergency, while providing additional safety to the platoon.

3 Model

This chapter will detail the system and attacker models of our simulations.

3.1 System Model

We will introduce the system model in this section.

A platoon is comprised of vehicles which are interconnected through a VANET, working together to improve traffic flow and fuel efficiency by increasing cohesion between nodes to reduce inter-vehicle distances and prevent unnecessary accelerations and decelerations. A central Certificate Authority (CA) issues pseudonymous certificates and public keys for participating vehicles. Each vehicle is controlled locally by a control algorithm. Participating nodes communicate according to the IEEE 802.11p standard, and communications are not impaired. Beaconing messages containing position, speed, acceleration, and a time stamp are sent to all adjacent vehicles at a rate of 10 Hz. During communication, sending nodes will provide their certificate along with their messages. Receiving nodes can then request the corresponding public key from trusted entities of the Public Key Infrastructure (PKI) to verify the certificate and by extension the identity of the sending node. The platoon

Our scenario will encompass a preexisting (i.e. nodes do not have to join the first) platoon of eight vehicles travelling on a straight highway segment with no additional traffic. Different scenarios target a platoon speed of 50, 100 and 150 km/h. We will investigate effects on three different CACC controllers. The first controller is a simple constant spacing controller as detailed in [21], which relies on data from common ACC sensors and the acceleration and speed values of the preceeding and leading vehicles to maintain a defined inter-vehicle distance, which we set to 5 and 20 meters for distinct runs. Another controller is the Ploeg controller [20] which extends the constant spacing CACC controller to gracefully degrade to ACC by estimating the speed and acceleration of the preceeding vehicle using on board sensors instead of merely relying on the ACC components of the control strategy in case of intermittent communication impairments like packet losses, as filtering messages would produce. The third controller implements a distributed consensus algorithm [1]. Each node incorporates position, velocity, and acceleration of all vehicles in the platoon to determine individual control inputs.

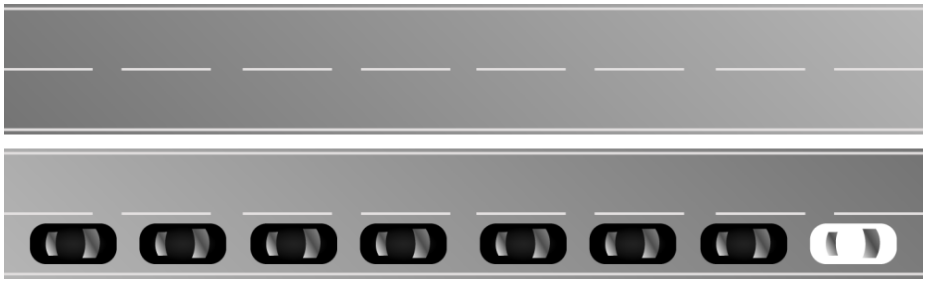


Figure 3.1: Platoon schematic. White vehicle is the leading node.

3.2 Attacker Models

This section will examine the different possible attacks towards vehicle platoons, and justify the scope of our simulations. Three general categories of attacks on CACC applications can be identified based on their point of attack. We do not consider privacy attacks, since they do not have a direct effect on platoon stability. In all cases, we assume robust software implementations and protocols which can not be exploited, i.e. have to be adhered to. The malicious node should be a node which is already a member of the platoon, but not the leader which has additional control over following nodes. The main objective of the attacker is to actively force a crash within the platoon or to otherwise impair the platoon [15].

3.2.1 Denial of Service or Network Layer attacks

Jamming, or Denial of Service (DoS) attacks aim to disrupt communications, denying applications information flow. Some component of a communication channel is chosen as the target of an attack, and the attack focuses on overwhelming the component. A successful attack results in loss or irrecoverable delay of vital information, leading to malfunctioning services and applications. Denying service to a CACC application can be realized through jamming of the 802.11p frequency band for example. A Hardware Security Module (HSM) which implements hashing algorithms or verifies/signs messages can present another weak point. Such cryptographic operations are often complex, and require relatively high process times, therefore placing a low upper bound on executable HSM operations per time. Targeting this limitation can lead to packet loss due to messages not being able to be verified in time. All Denial of Service attacks prevent a timely delivery of messages, preventing any communications-based controller from working correctly, and can not be detected or mitigated using semantic filterings. They are therefore not relevant to this work.

3.2.2 Sensor Manipulation or Hardware Layer attacks

The second category of attacks encompasses any attack which attempts to alter data from the car's internal sensors as they are passed to the controller. Performing such an attack requires the attacker to have direct access to the vehicle's hardware, and can be realized by altering data which is written to a bus system or transmitted through an internal network before the controller reads it, or through a software exploit which allows access to the controller's memory where the data can be changed before it is accessed by the controller. This kind of attack can produce similar symptoms to an Application Layer attack, but the software attack especially is much harder to detect and prevent. Another way to attack vehicle sensors is through external influences. External sensors observe the environment, and wave-based sensing technologies especially can be ”blinded” [18]. Verifying data of a sensor and detecting malfunctions or malicious tampering programatically is a complicated task without information about the current situation.

3.2.3 Application Layer attacks

Application Layer attacks are the third, and for this work most interesting category of attacks. An Application Layer attack is any attack where an attacker, or compromised entity sends altered information with malicious intent, i.e. to destabilize the platoon or provoke a crash. The category can be further divided into spoofing/replay, message falsification/data injection.

In a spoofing or replay attack, a malicious entitiy records network traffic of legitimate entities, and uses the information gathered to impersonate another node (spoofing) or re-send outdated beacons (replay) in order to harm the platoon. In single-hop networks for example, an adversary can target a specific node by impersonating its preceeding vehicle and force it to slow down until the platoon has to be dissolved. Spoofing attacks are largely avertable through cryptographic signing of messages, while replay attacks can be prevented through clock synchronization, and thus do not usually need to be filtered through semantic filtering.

In message falsification or injection attacks, an adversary node will either send beaconing messages with incorrect information about its circumstances so that the platoon is influenced unfavorably, or intercept messages of another node, and alter the information therein. Among these categories of attacks, this type is most likely to be detectable through semantic filtering. Depending on the CACC controller, different properties might be communicated among cars, and the final manifestation of the attack changes accordingly. The severity of the consequences of a message falsification relate to the altered properties, as well as the magnitude of the changes, i.e. tampering with acceleration values might cause more grave consequences than position data.

For this work, we focus on message falsification attacks. We will use a simulation setup similar to [10], extending it to include message filtering, but omitting the jamming attack scenario, seeing as it is not suited to examine the capabilities of our misbehavior detection approach. This leaves us with three data injection scenarios, altering position, speed and acceleration respectively, built on top of the ”Sinusoidal” scenario of Plexe, which was originally made to examine platoon behavior. For the position injection scenario, the reported position is made to drift away linearly from the actual position after the attack starts, while the speed and acceleration injection attacks report a static value for their respective property rather than the actual value.

platoon size	8
attacker ID	3
controllers	CACC, CACC, Ploeg, Consensus
spacings	5, 20, 20, 20
target speeds	50, 100, 150 km/h
position attack values	3, 5, 7, 9, 11 m/s
speed attack values	-50, 0, 50, 100, 150 km/h
acceleration attack values	-30, -10, 0, 10, 30 m/s^2

Table 3.1: attacker models

4 Implementation

In order to find out how semantic data filters could be implemented into simulations and affect ride safety, two things are necessary.

First, we need a simulation framework which supports both road and network traffic simulations. Plexe offers easy extensibility and allows us to compare directly to previous work.

Rather than implementing detection mechanisms into Plexe, the Maat framework offers several detectors which consider new data from incoming messages with regards to the current situation, and return a rating of the message's trustworthiness. The results of the individual detectors are then fusioned to generate opinions on the messages regarding different metrics.

4.1 Plexe

Plexe [23] is an extension of the Veins simulation framework. Like Veins, it is built on top of OMNeT++, and interfaces with an adapted version of SUMo. Plexe-SUMo can incorporate data from inter-vehicle communications into road traffic simulations, allowing for the formation and maintenance of platoons of vehicles. We used version 2.1 for our simulations. In Plexe, each entity is an application, which extends upon a base application. The base application only relays information like vehicle position and speed from platooning beacons back to SUMo. Additional functionalities like attacker models, different modes of data output, or communication with other software are implemented as extensions upon the base application.

4.2 Maat

Maat is an extensible Java framework for misbehavior detection. It provides a data model for detection algorithms to work on, as well as several detection algorithms out of the box. Maat consumes V2V communication messages in JSON format and applies them to a world model. The changes each message implies are rated based on possibility and probability according to different detectors with each one focusing on different aspects like distance moved, speed or sudden appearance. The output of all active detectors is combined using a fusion algorithm [25], forming an opinion on the overall trustworthiness of the original message in real time. We chose to enable the Sudden

Appearance Warning (SAW), Acceptance Range Threshold (ART), Distance Moved Verifier (DMV) and Simple Speed Check (SSC) detectors in our simulations. The Sudden Appearance Warning detector is suited to filter messages of new neighbors, claiming to appear very close to a node. Initial beacons from nodes claiming to be very close (below a threshold) to the observing node are deemed untrustworthy. For the Acceptance Range Threshold detector, the opposite is true. Beacons placing a node well within the transmission range (supplied through a threshold parameter) score higher values for trustworthiness. The Distance Moved Verifier simply ensures that vehicles move a minimum distance between beacons, effectively scoring repeat transmissions of same positions with low trustworthyness. The Simple Speed Check detector is heavily inspired by Kalman filters, extrapolating the position of neighbor nodes based on previous speed and position beacons, and updating predictions based on newer beacons. Opinions are computed from velocity differences of two messages. Resulting opinions from these detectors are fusioned using the Weighted Belief Fusion algorithm. The detectors as well as the WBF fusion algorithm are further detailed in [8, ch 5.2.4, ch 6.1] respectively. We came up with two different strategies to binarize the fusioned opinions. For the first strategy, we binarize based on "Belief/Disbelief", effectively making the decision on whether or not a message should be dismissed by naively comparing the belief and disbelief properties of the opinion. The alternative strategy is based on the expected opinion, which projects belief and uncertainty to a probability $P_X(x)$ by distributing the uncertainty over the potential values in X according to the base rate a(x) as

$$P_X(x) = b_X(x) + a_X(x) \cdot u_X$$

If this "Expectation" is greater than 0.5, the message is deemed sufficiently believable with acceptable certainty.

detectors	eSAW, eART, eDMV, eSSC
parameters	200, 500, 1, 5
fusion algorithm	Weighted Belief Fusion
opinion binarization methods	Belief/Disbelief, Expectation

Table 4.1: detector parameters

4.3 Accomodations

The data injection attacker models of van der Heijden have already been implemented for an older version of Plexe and could easily be reused with minor adjustments to the code and simulation configurations to fit the newer base version of Plexe.

As a first step to allow communications between Plexe and Maat, a message format has to be chosen. The default output format for Plexe is vector based. In the header sec-

tion of such an output file, each property of each node is assigned a vectorID. Whenever a property changes, a line in the form of

$$vectorID \quad eventID \quad timestamp \quad value$$

is added to the output. Maat, however expects messages in a simple JSON-format, and does not require every possible property to be present. For any message, only a time-of-reception as well as a message type have to be present. Beacon messages further require fields to identify the sender as well as a message ID. In order to be able to do meaningful filtering, position, speed and transmission times should also be supplied, but the information required ultimately depends on the enabled detectors. The JSON format appears to be a much better solution for our use case, based on the effort required to implement widely used JSON output compared to parsing the existing vector format of Plexe. Data transmission will be handled through TCP sockets, with Maat offering the service, thus implementing the server side. Switching from file to socket input in Maat is done through the $-s$ parameter, followed by a port to listen on. Supplying the switch and an unused port will open a $ServerSocket$ to listen for incoming connections, and switch from a line-based input reader using $BufferedReader$ to a more robust version suited for TCP connections using $DataInputStream$. Each node in Plexe corresponds to a Maat instance, building the known world model for that node and forming opinions for the node based on incoming messages from other nodes. Switching communications with Maat on or off in Plexe is done through setting the $useMaatSocket$ parameter of each node's application in the simulation configuration file. The Plexe application determines the port of the corresponding Maat instance as $baseport + nodeID$, where the base port is currently hard-coded. In order to make sure we only read full messages on the Maat side, each JSON message is prefaced with four bytes containing the message length when sent from Plexe. Full messages are then handled: parsed for different properties like position and speed, and the information is committed to the world model. The differentials from comitting new information serve as input to the active detectors, which produce a rating for the message in the form of multiple opinions, which are then fusioned into a singular subjective opinion using the Weighted Belief Fusion method dubbed ”the best choice overall” in [8, ch. 9]. We binarize fusioned subjective opinions using two different strategies in different runs. The binarized opinion is transmitted back to Plexe as a single byte message. This is done to avoid having to write out the opinion, and parsing it again in Plexe.

All source and configuration files are made available on GitLab for further details/reference.

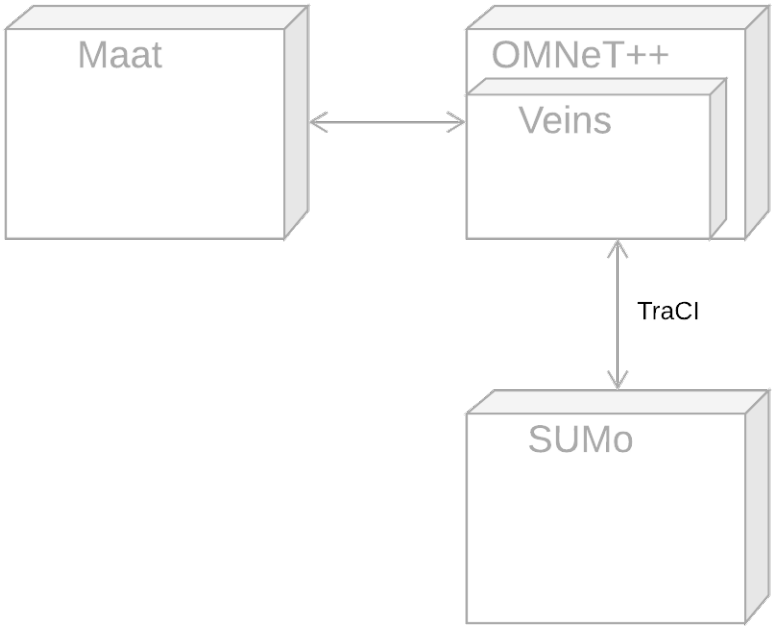


Figure 4.1: model view of the experimental setup

5 Evaluation

This chapter is dedicated to the selection of metrics to quantify the success of the misbehavior detection, and to present the results of our simulations.

5.1 Metrics

Since our results will relate to previous work, it seems wise to adopt similar metrics. Success of a malicious attack can best be assessed in a binary fashion, distinguishing between success and no success. In the case of success, or in our case a crash, different scenarios can be compared further based on the severity of the crash. Medical research ([13], [12]) suggests impact velocity Δv as an adequate indicator. If no crash occurs, the platoon may still be disrupted, which may manifest as a deviation from the desired inter-vehicle distance. We choose the maximum spacing error $max_{s,i,t}(e)$, where s is the simulation run, i is the i-th vehicle on the platoon, t is time and e is the absolute spacing error, as significant metric, because it best represents the closeness to success (i.e. a crash), rather than the average maximum spacing error $avg_s(avg_i(max_t(e)))$ or average maximum acceleration $avg_s(avg_i(max_t(a)))$ which better correlate to perceived ride quality for passengers [14] but are less relevant to our research objective.

5.2 Results

This section presents the results we obtained from running each scenario, using the parameters listed in tables 3.1 and 4.1. We re-ran the original simulations without additional misbehavior detection active on our updated experimental setup to ensure that changes to the simulator would not influence our evaluations. For each injection attack, we did this run of the original simulation configuration to generate data we could compare to, and then ran the simulations two more times with misbehavior detectors enabled and different decision making strategies. We hope to find evidence that data-centric misbehavior detection is indeed an effective strategy to prevent negative effects of data injection attacks in platooning scenarios. A decrease in crashing scenarios of runs incorporating misbehavior detection compared to those runs without misbehavior detection would be desirable in this regard, indicating that semantic data filters are indeed effective.

5.2.1 Position Injection

The base data generated from simulations without misbehavior detection shows (in figure 5.1a), that our shifting position is only effective against the Consensus-based controller. Platoon target speed appears to affect the velocity difference of the crash, albeit only minorly. The constant spacing CACC controllers appear resilient, most likely due to them not factoring positional data into the calculation of their desired acceleration. Unrelated to the attack, it is noteworthy that the Ploeg controller's spacing error scales with the platoon's target speed. The consensus controller produced crashes regardless of attack value, and the impact velocity appears only to be linked to the leader speed, unlike in [10], where a relation between attack value and impact velocity could be derived.

Adding our filters using the naive Belief/Disbelief decision making strategy does not change the results at all (see figure 5.1b). The Consensus controller exhibits the same sensitivity towards the attack, and the constant spacing CACC and Ploeg controllers do not change behavior either. This has to be attributed to the fact that our decision making strategy deemed none of the messages malicious enough to be dropped, and network traffic was therefore unchanged.

Utilizing expected opinions to determine malicious messages presents an identical pattern to the previous two runs of the position injection attack, signifying that this strategy did not perform as we hoped either. The number of filtered messages from each run attests to this lack of efficacy, showing 0 filtered messages.

5.2.2 Speed Injection

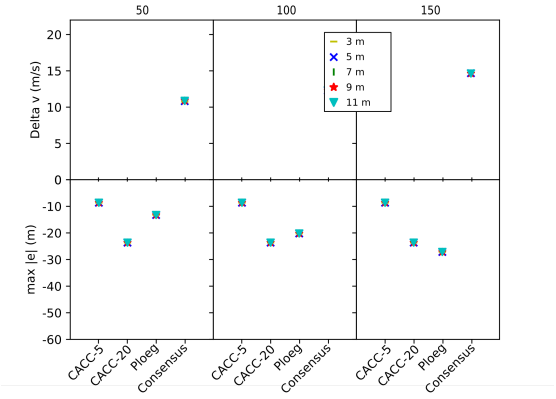
For our speed injection attacks, instead of shifting the reported value based on time passed since attack start, the attacker instead continuously reports the attack value.

Our base case results (figure 5.2a) without misbehavior detection see the constant spacing controllers producing crashes for all platoon speeds, except for cases where attack value and platoon speed coincide. In all cases where a crash occurs, lower attack values imply greater Δv, and a relation between target speed and impact velocity is apparent. The Ploeg and Consensus controllers appear to be resilient to the attack, but spacing errors relate to the target leader speed.

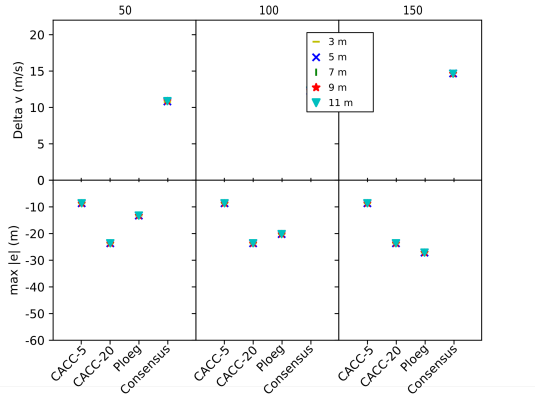
Runs using the Belief/Disbelief and Expectation strategy (5.2b, 5.2c) for misbehavior detection still do not show any differences to their unfiltered counterparts, and statistics on filtered messages support this again, by showing no filtered messages across all nodes of all runs.

5.2.3 Acceleration Injection

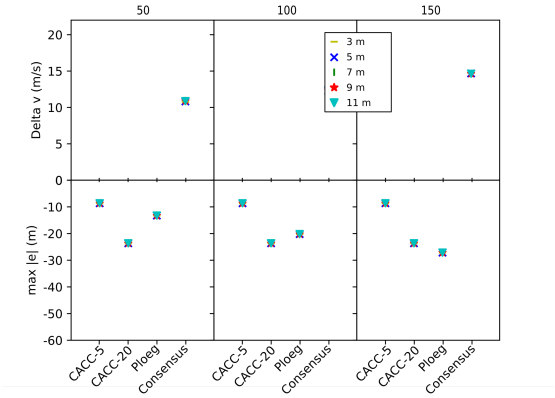
As for the speed injection attacks, instead of shifting its values, the attacker node in these runs simply reports the attack value, disregarding the situation. While the detec-



(a) Simulation results without additional misbehavior detection active

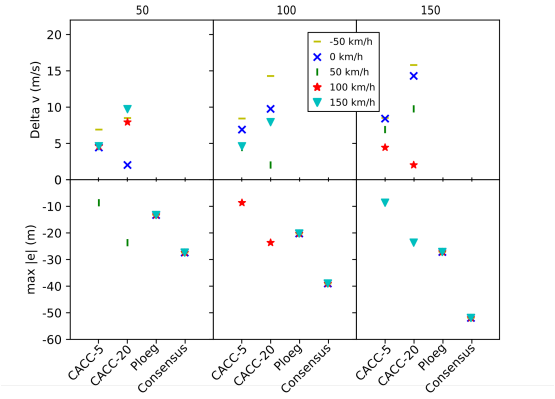


(b) Simulation results with misbehavior detection active. Binarization: Belief > Disbelief

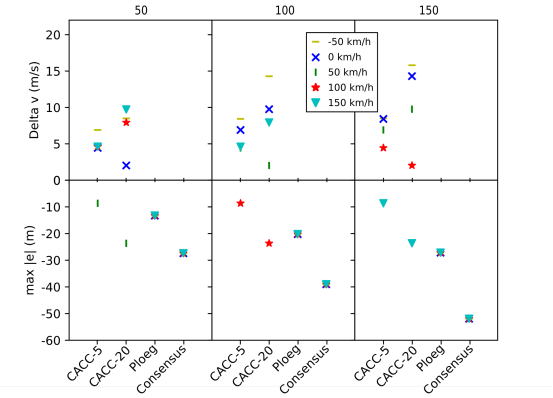


(c) Simulation results with misbehavior detection active. Binarization: Expectation > 0.5

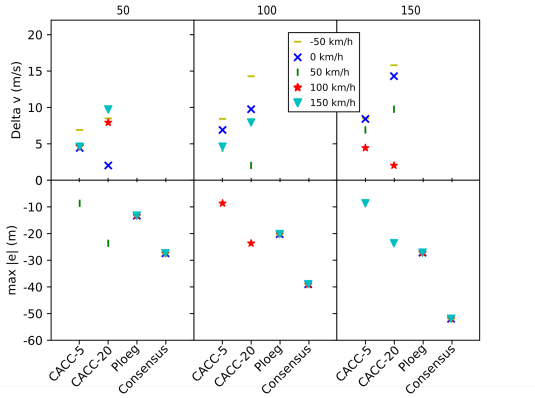
Figure 5.1: Position Injection Attack



(a) Simulation results without additional misbehavior detection active



(b) Simulation results with misbehavior detection active. Binarization: Belief > Disbelief



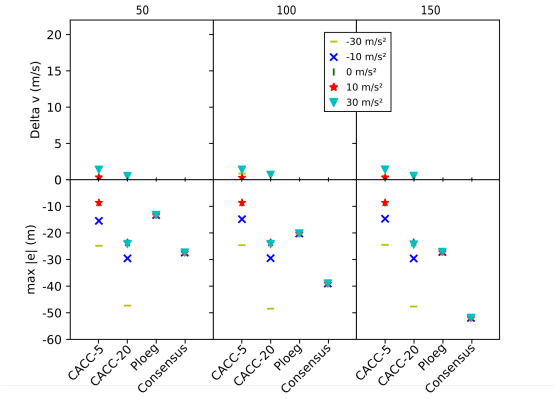
(c) Simulation results with misbehavior detection active. Binarization: Expectation > 0.5

Figure 5.2: Speed Injection Attack

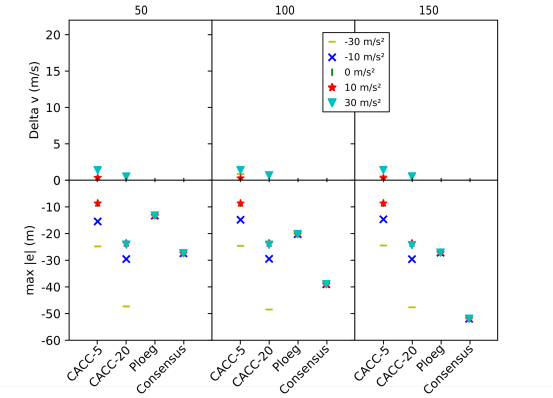
tion parameters are not suited to detect an attack of this kind, we included these runs for completeness.

The base case (depicted in figure 5.3a) once again shows the CACC-5 and CACC-20 controllers to have similar issues, as expected. No matter the target speed, attacking the CACC-5 controller with a false acceleration value of $30m/s^2$ led to a crash. It is worth to note, that the CACC-5 controller was just on the verge of crashing, having runs using the same parameters result in crashes while others did not. This behavior appeared regardless of target speed for the $10m/s^2$ attack and for the $-30m/s^2$ attack at the two higher leader speed values. Similar behavior was recorded for the CACC-20 runs, regardless of platoon velocity, for the $30m/s^2$ attack. Furthermore, CACC-20 exhibited a wider range of spacing errors than CACC-5 in cases where neither crashed. The Ploeg controller handles the attack well, and exposes the same relations between target speed and spacing error once again. The consensus based controller does not produce any crashes either, showing similar behavior to the Ploeg controller, albeit magnified by a factor of roughly 2.

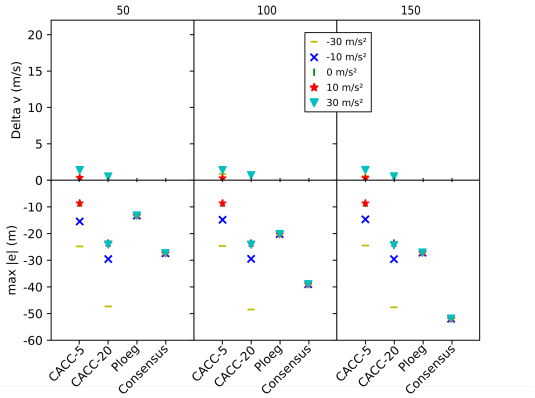
Simulations with misbehavior detection enabled and utilizing either the Belief/Disbelief or Expectation binarization once again do not actually detect the attack and are not able to filter any messages, therefore yielding the same results shown in figures 5.3b, 5.3c as the base acceleration attack case.



(a) Simulation results without additional misbehavior detection active



(b) Simulation results with misbehavior detection active. Binarization: Belief > Disbelief



(c) Simulation results with misbehavior detection active. Binarization: Expectation > 0.5

Figure 5.3: Acceleration Injection Attack

6 Discussion

Across all attacks, a similar pattern repeats itself. Our filtering misbehavior detection underperforms, effectively not doing any work and therefore yielding the same results as runs without enabled misbehavior detection.

6.1 Belief/Disbelief Based Decision Making

Naively comparing the belief and disbelief values of subjective opinions yielded underwhelming results. The data suggests, that no filtering took place at all. This can easily be attributed to the detector setup we used not being suited for the attack scenarios. During all runs using this strategy, no single message was deemed malicious. Using this strategy however can give valuable insight into the detection setup's ability to determine whether or not malicious messages were more trustworthy or untrustworthy. It could prove useful in trying to determine whether or not a specific detector is suited to detect a type of attack. Incorporating a threshold into the the decision making can give further insight into the confidence of the detection, beyond the uncertainty value.

6.2 Expectation Based Decision Making

Using the alternative strategy to elect untrustworthy messages based on expected opinions, the results of all three attacker models still remained unchanged from the base case which did not incorporate our misbehavior detection mechanisms. This strategy based on expected opinions has a higher tendency compared to the Belief/Disbelief comparison, to allow messages which yielded opinions with high uncertainty after fusion. Despite being just as inefficient as the alternative strategy for our use case, we deem this a more sane approach. In a hypothetical scenario where detectors need a warm-up period, during which belief and disbelief values are fluctuating around low values, this approach could help to reduce resulting jitters which, using our filtering approach would manifest as intermittent packet losses, depriving the active controllers of possibly important data. For messages yielding more certain opinions, close decisions will be made with less lenience.

6.3 Detectors

Seeing how our chosen detectors and parameters were unsuccessful in detecting even the position injection attacks which we considered to be most likely to be detected based on previous results [9], it stands to reason that our choice of detection parameters either weren't suited for the attacks, or that the detection algorithms themselves require further improvements.

7 Conclusion

In this paper we attempted to prove the efficacy of data-centric misbehavior detection using message filters in a platooning scenario. We employed attacker models which had been proven to be effective against platoons before, and attempted to mitigate those effects using message filters. Even though we were not able to confirm positive effects using our experimental setup, we were able to connect simulations and real time misbehavior detection, and hopefully develop a base for further experimentation.

The issue of high uncertainty discussed in chapter 6 motivates experiments using different detector parameters to discern whether or not the detectors are suited to uncover our attacker models at all. Further experimentation using different attack values could be employed to prove this aswell.

We also gained insight on the suitability of different binarization or decision making strategies. Using the discrepancy between belief and disbelief as the predicative metric may not be the optimal approach for combined detection efforts. Instead, it might prove more useful when trying to determine whether or not a detector is prone to being able to correctly detect an attack type, where uncertainty could be less of a focus. The strategy could also employ a threshold to filter out jitters for decisions where belief and disbelief are very close to one another. Expected opinions employ both, belief and uncertainty values to produce behavior similar to a dynamic thresholded belief/disbelief comparison. We think it is a good idea to consider both concepts when analyzing detectors, or combinations of such.

Bibliography

[1] M. di Bernardo, A. Salvi, and S. Santini. “Distributed Consensus Strategy for Platooning of Vehicles in the Presence of Time-Varying Heterogeneous Communication Delays”. In: *IEEE Transactions on Intelligent Transportation Systems* 16.1 (Feb. 2015), pp. 102–112.

[2] F. Boeira, M. P. Barcellos, E. P. de Freitas, A. Vinel, and M. Asplund. “Effects of colluding Sybil nodes in message falsification attacks for vehicular platooning”. In: *2017 IEEE Vehicular Networking Conference (VNC)* (Nov. 2017).

[3] S. Dadras, R. M. Gerdes, and R. Sharma. “Vehicular Platooning in an Adversarial Environment”. In: *Proceedings of the 10th ACM Symposium on Information, Computer and Communications Security - ASIA CCS ’15* (2015).

[4] R. G. Engoulou, M. Bellaïche, S. Pierre, and A. Quintero. “VANET security surveys”. In: *Computer Communications* 44 (May 2014), pp. 1–13.

[5] J. Erickson, S. Chen, M. Savich, S. Hu, and Z. M. Mao. “CommPact: Evaluating the Feasibility of Autonomous Vehicle Contracts”. In: *2018 IEEE Vehicular Networking Conference (VNC)* (Dec. 2018).

[6] H. Hartenstein and K. Laberteaux. *VANET: vehicular applications and inter-networking technologies*. Vol. 1. Wiley Online Library, 2010.

[7] R. W. van der Heijden, S. Dietzel, T. Leinmüller, and F. Kargl. “Survey on Misbehavior Detection in Cooperative Intelligent Transportation Systems”. In: *IEEE Communications Surveys & Tutorials* (2018), pp. 1–1.

[8] R. W. van der Heijden. “Misbehavior detection in cooperative intelligent transport systems”. PhD thesis. http://dx.doi.org/10.18725/OPARU-11089: Ulm University, 2018.

[9] R. W. van der Heijden, T. Lukaseder, and F. Kargl. “VeReMi: A Dataset for Comparable Evaluation of Misbehavior Detection in VANETs”. In: *CoRR* abs/1804.06701 (2018).

[10] R. van der Heijden, T. Lukaseder, and F. Kargl. “Analyzing attacks on cooperative adaptive cruise control (CACC)”. In: *2017 IEEE Vehicular Networking Conference (VNC)* (Nov. 2017).

[11] D. Jiang and L. Delgrossi. “IEEE 802.11 p: Towards an international standard for wireless access in vehicular environments”. In: *VTC Spring 2008-IEEE Vehicular Technology Conference*. IEEE. 2008, pp. 2036–2040.

[12] H. C. Joksch. “Velocity change and fatality risk in a crash–a rule of thumb”. In: *Accident Analysis & Prevention* 25.HS-042 059 (1993).

[13] C. Jurewicz, A. Sobhani, J. Woolley, J. Dutschke, and B. Corben. “Exploration of Vehicle Impact Speed – Injury Severity Relationships for Application in Safer Road Design”. In: *Transportation Research Procedia* 14 (2016), pp. 4247–4256.

[14] M. Larburu, J. Sanchez, and D. J. Rodriguez. “Safe road trains for environment: Human factors’ aspects in dual mode transport systems”. In: *ITS World Congress, Busan, Korea*. 2010, pp. 1–12.

[15] J.-P. Monteuuis. “Attacker model for Connected and Automated Vehicles”. In: 2018.

[16] S. Noei, A. Sargolzaei, A. Abbaspour, and K. Yen. “A Decision Support System for Improving Resiliency of Cooperative Adaptive Cruise Control Systems”. In: *Procedia Computer Science* 95 (2016), pp. 489–496.

[17] J. Petit, M. Feiri, and F. Kargl. “Spoofed data detection in VANETs using dynamic thresholds”. In: *2011 IEEE Vehicular Networking Conference (VNC)* (Nov. 2011).

[18] J. Petit, B. Stottelaar, M. Feiri, and F. Kargl. “Remote attacks on automated vehicles sensors: Experiments on camera and lidar”. In: *Black Hat Europe* 11 (2015), p. 2015.

[19] A. Petrillo, A. Pescape, and S. Santini. “A collaborative control strategy for platoons of autonomous vehicles in the presence of message falsification attacks”. In: *2017 5th IEEE International Conference on Models and Technologies for Intelligent Transportation Systems (MT-ITS)* (June 2017).

[20] J. Ploeg, E. Semsar-Kazerooni, G. Lijster, N. van de Wouw, and H. Nijmeijer. “Graceful degradation of CACC performance subject to unreliable wireless communication”. In: *16th International IEEE Conference on Intelligent Transportation Systems (ITSC 2013)* (Oct. 2013).

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

[22] M. Saini, A. Alelaiwi, and A. E. Saddik. “How Close Are We to Realizing a Pragmatic VANET Solution? A Meta-Survey”. In: *ACM Comput. Surv.* 48.2 (Nov. 2015), 29:1–29:40.

[23] M. Segata, S. Joerer, B. Bloessl, C. Sommer, F. Dressler, and R. L. Cigno. “Plexe: A platooning extension for Veins”. In: *2014 IEEE Vehicular Networking Conference (VNC)* (Dec. 2014).

[24] M. A. Shahid, A. Jaekel, C. Ezeife, Q. Al-Ajmi, and I. Saini. “Review of potential security attacks in VANET”. In: *2018 Majan International Conference (MIC)* (Mar. 2018).

[25] R. W. Van Der Heijden, H. Kopp, and F. Kargl. “Multi-Source Fusion Operations in Subjective Logic”. In: *2018 21st International Conference on Information Fusion (FUSION)* (July 2018).

[26] W. Whyte, A. Weimerskirch, V. Kumar, and T. Hehn. “A security credential management system for V2V communications”. In: *2013 IEEE Vehicular Networking Conference* (Dec. 2013).