UNIVERSITY OF CALIFORNIA
RIVERSIDE

Secure Automated and Autonomous Systems

A Dissertation submitted in partial satisfaction
of the requirements for the degree of

Doctor of Philosophy

in

Electrical Engineering

by

Ahmed H. Abdo

September 2022

Dissertation Committee:

Professor Nael Abu-Ghazaleh, Chairperson
Professor Matt Barth
Professor Hyoseung Kim
Professor Daniel Wong

The Dissertation of Ahmed H. Abdo is approved:

_____

_____

_____

_____

Committee Chairperson

University of California, Riverside

Acknowledgments

Thank you, Almighty Allah, the most gracious and merciful, for blessing me during this journey. I would like to thank the University of California at Riverside (UCR) for sponsoring my Ph.D. study. It would not have been possible to write this doctoral dissertation without the help and support of the amazing people around me. I want to thank each of them individually for making this happen. It would not have been possible to write this doctoral dissertation without the help and support of the fantastic people around me. I want to thank each of them individually for finally making this happen!

First, I would like to express my profound appreciation to my academic advisor, professor Nael Abu-Ghazaleh. He always believed in me, supported me in my professional and personal lives, and guided and mentored me during this educational journey. He is and will always remain my best role model as a scientist, mentor, and teacher. It is truly my great pleasure and honor to know him.

I thank my committee members, professor Matt Barth, professor Hyoseung Kim, and professor Daniel Wong. They generously offered their time, support, guidance, and goodwill throughout the preparation and review of this dissertation. I want to express my thankfulness to Dr.Guoyuan Wu for all the support he showed me during my Ph.D. He is a fantastic person, instructor, scientist, and friend. I want to thank Basem Mahmoud, Mohamed Refaat, Abdulrahman Fahim, and Mohamed Sedki, who rallied around me during my illness. Their significant support helped me during my recovery.

It was my honor to work with the best minds in the field, my collaborators, Dr.Qi Zhu, Dr.Zhiyun Qian, Xishun Liao, Xuanpeng Zhao, and Sakib Md Bin Malek. Without their collaborations, I would never finish and published my work in the past few years. I also would like to thank my wonderful lab mates: Khaled Khasawneh, Hoda Naghibijouybari,

Jason Zellmer, Fatemah Elharbi, Hodjat Asghari, Abdulrahman Bin Rabiah, Shafiur Rahman, Shirin HajiAminShirazi, Sakib Md Bin Malek, Esmaeil (Reza) Mohammadian Koruyeh, Bradley Evans, Ashay, Shirwadkar, and Sankha Dutta. Their support and kindness greatly enhanced my experience in the lab.

Also, I would like to thank my childhood friends back home, in Egypt, and here, in the US, for all their love and support. I am thankful our paths have crossed. This journey would not have been possible without the support of my beloved parents. I always hope they are proud of me, and may Almighty Allah protect you. In addition to my siblings, parents-in-law, my lovely nephews, and nieces. Throughout my life, their love and support have enabled me to pursue what truly interests me and have made me the person I am today.

Finally, and most importantly, to my lovely wife: you are my life! Your contribution is invaluable; no words can express how thankful I am that you are my wife and best friend. You sacrificed a lot to help me accomplish my childhood dream—to get this Ph.D. Without your support, none of this work would have seen the light of the day.

ABSTRACT OF THE DISSERTATION

Secure Automated and Autonomous Systems

by

Ahmed H. Abdo

Doctor of Philosophy, Graduate Program in Electrical Engineering
University of California, Riverside, September 2022
Professor Nael Abu-Ghazaleh, Chairperson

Automated and autonomous systems are emerging new technologies that promise to revolutionize transportation and traffic applications. Connected vehicles (CV) applications can improve safety, efficiency, and capacity of transportation systems while reducing their environmental footprints. A large number of CV applications have been proposed towards these goals, with the US Department of Transportation (US DOT) recently initiating three deployment sites. Unfortunately, the security of these protocols has not been considered carefully, and due to the fact that they affect the control of vehicles, vulnerabilities can lead to breakdowns in safety (causing accidents), performance (causing congestion and reducing capacity), or fairness (vehicles cheating the intersection management system). In our work, we perform a detailed analysis of a recently published CV-based application protocol, Cooperative Adaptive Cruise Control (CACC), and use this analysis to classify the types of vulnerabilities that occur in the context of connected Cyber-physical systems such as CV. We show, using simulations, that these attacks can be extremely dangerous: we illustrate attacks that can cause crashes or stall emergency vehicles. We also carry out a more systematic analysis of the impact of the attacks, showing that an individual attacker can even have substantial effects on traffic flow and safety, even in the presence of message security standard developed by US DOT. We believe that these attacks

can be carried over to other CV applications if they are not carefully designed. We also explore various defense frameworks to mitigate these classes of vulnerabilities in CV applications. At the same time, autonomous systems AVs are vulnerable to physical attacks that manipulate their sensors through spoofing or other adversarial inputs. If the sensor values are incorrect, an autonomous system that acts on them can be made to malfunction or even controlled to perform an adversary's chosen actions, making this a critical threat to the success of these systems. To counter these attacks, recent works propose developing physics- based detectors that estimate the future state of an autonomous vehicle and use this estimate to detect anomalous sensor inputs. The accuracy and responsiveness of this detection algorithm are important to the security and robustness of autonomous systems. State of the art solutions that are based on Kalman filters face challenges in terms of configuration parameters and the limitations of the algorithm: we show that, while they constrain some attacks, an attacker is still able to bypass them. We focus on the security of CVs and AVs in terms of application- level attacks and defenses. First, we demonstrate scenarios where the vulnerabilities can be exploited to cause safety breakdowns or to interfere with an emergency vehicle. We define metrics for evaluating the attack impact that measures mobility (traffic throughput) and safety (average separation between cars). We show that attacks can substantially interfere with the operation of CVs, leading to increased vehicular speeds and reduced safety margins. Having established these attacks on the CVs application-level, we need to consider a mitigation framework where we use the classification of the five vulnerability types we introduce to guide the design of the mitigation steps that either eliminate or interfere with them. For example, one class of vulnerabilities occurs when the application logic does not check whether the data in the messages are consistent with other information it has about the system. Some of the message values are impossible to verify due to the lack of independent information to confirm it. Thus, we propose having an alternative source of data (specifically, data from cameras) to validate information in CVs application messages. We show

that the defense indeed mitigates the vulnerabilities we identified in CVs without substantially harming performance. We also use *blockchain*, which is traditionally used in applications from cryptocurrencies to smart contracts, as a potential solution to CV security. The BlockChain technology has the potential to revolutionize connected vehicles. It is far more secure than other record keeping systems because each new message transaction is encrypted and linked to the previous transaction. Specifically, we exploit the immutability of BlockChain to ensure safety from falsified information and attacks. Therefore, we propose a BlockChain -based scheme to protect the vehicular ecosystem and increase its security. We demonstrate these properties by developing an algorithm that uses BlockChain to maintain trusted communications between vehicles in the context of a cooperative ramp merging application. Next, we propose a new system to defend against physical attacks on AVs through: (1) Training the Kalman Filter to improve its ability to operate within its target environment; (2) Introducing a residual machine learning- based algorithm to capture non-linear dynamics of the system; and (3) Incorporating a change detection model to detect anomalies in the temporal behavior of the sensor data, to improve the assessment of deviation between the predicted and measured data. Our framework combines components that track a number of non-linear physical invariants and derives additional learned invariants coefficients through an optimization algorithm. It also uses an optimized residual prediction module based on a neural network, followed by a change detection algorithm, for keeping track of the historical anomalies. Taken together, these ideas enable for high accuracy to estimate the physical state of the vehicle, detecting a number of attacks that bypass state of the art defenses, with low overhead compatible with real-time implementations.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

The automotive industry [68] will introduce the autonomous vehicle into the consumer market in the near future. At this moment, several forms of connected vehicles have already been introduced. Both types of vehicles provide the foundation to propel the automotive industry into the technological future. The United States Department of Transportation (US DOT) has been developing next-generation Intelligent Transportation Systems (ITS) [1] where vehicles and transportation infrastructure communicate and collaborate towards goals, such as improving safety, increasing traffic flow capacity, supporting driver assistance functionality, and reducing overall carbon footprint [25]. Some of these technologies are already installed across the country and can be found in traffic signal coordination, transit signal priority, and traveler information systems.

During the initial stages when researchers and engineers are developing early prototypes of connected vehicles and autonomous applications, security is not being considered deeply. For example, connected vehicles expose a large attack surface with many participants and complex functionality. Attacks may target application protocols, networking, and sensing and vehicle control, with the potential to cause accidents, traffic delays and other harm to the system. The

US DOT utilizes a Secure Certificate Management System (SCMS) to ensure that cars and road side units have certificates which enable them to participate in communication [45].

Autonomous vehicles [94] also suffer from two great threats to their sensors: (1) GPS spoofing and (2) transduction attacks. GPS spoofing attacks have occurred in real-world systems and can badly affect autonomous vehicles via their navigation systems. Other important attacks against autonomous vehicles are transduction attacks, which inject out-of-band signals to sensors or actuators in order to translate a physical signal into an electrical one. However, these sensors can sometimes pair the property being measured and the analog signal, which can be manipulated by the attacker. For example, sound waves can affect accelerometers and make them report incorrect movement values, while radio waves can trick pacemakers into disabling pacing shocks. These attacks have been shown to be effective on autonomous vehicles by using sound to affect gyroscopes, lasers to affect camera sensors in drones, lasers to affect lidar sensors in cars, and intentional electromagnetic interference to manipulate actuators in drones.

1.1 Motivation

Connected and autonomous vehicles [53] are considered "phones with wheels" and combine cars, telecommunications, and IT risks while centralizing them in car fleet management. Notably, phones are some of the most potent surveillance devices ever made, and thus, they expose cars to hazards related to surveillance, privacy, and fraud. Now more than ever, it is essential to explore these risks to better prepare the proper defenses against threats and future-proof the security of the connected and autonomous vehicles. This dissertation aims to help understand cyber security on the single- vehicle level. It also evaluates the cyber-security problem at a multi-vehicle level and transportation system level. In order to secure any automotive system, we first need to ensure that it is designed and developed to enable vehicles to have confidence in

one another and their sensors. Thus, we must study the different vehicular credential management systems and evaluate their reliability in various scenarios and cases. We also need to research application layer attacks affecting vehicle applications' functionalities since these attacks are particularly dangerous and critical and not well studied. We also need to look at different vehicular cyber defenses, study their efficiencies, and design better versions with improved accuracies and run times. As a result, this dissertation is motivated by three security aspects that are very critical for connected and autonomous systems: 1) Automated applications and authentication vulnerabilities, 2) Physical based attack detection and 3) Anomaly detection.

1.1.1 Automated applications and authentication vulnerabilities

CV applications rely on the use of the Security Credentials Management System (SCMS) [5], and most recently, the US DOT deployed SCMS devices at three CV pilot sites (New York, Tampa, and Wyoming) [9, 10, 8]. The current implementation is a proof-of-concept certificate-based authentication system which uses a Public Key Infrastructure [45] for certificate management. Pseudonym Certificates (PCs) are used and rotated to enable message authentication and validation without exposing the privacy of a vehicle by having a permanent certificate. A CV can enroll in the system by submitting an enrollment request to US DOT. A PC can be obtained by vehicles for short-term use, ranging from 5 minutes to few days, and is utilized for basic safety message (BSM) authentication. On Board Equipment (OBE) uses identification certificates to authenticate itself in V2I applications. However, none of the V2I applications we reviewed require encryption by the OBE at the application level. SCMS prevents an attacker from falsifying messages from another vehicle, as each message gets signed with a certificate. However, SCMS cannot prevent a malicious actor from obtaining a certificate and participating in the protocol by replaying the messages while they are valid, or sending its own message, with fabricated data, using its certificate. Although the protocol discusses detection of potential

malicious activities and reporting them to revoke certificates ostensibly, this behavior is currently completely undefined. It is important to note that the SCMS certificate protocol prevents a number of attacks. Without it, an attacker can merely spoof any other car, and interfere with any maneuver. In contrast, with this protection in place, the attacker may only replay messages, or send bad information using its certificate. In general, application- level exploitation is possible, and perhaps can be used in conjunction with application- level attacks to amplify their damage. This could be seen in situations where the attacker is a compromised vehicle which uses a radio capable of reaching cars farther away than typical vehicular radios, and is capable of authenticating itself to the SCMS as a regular vehicle, then applying its attacks in the application level.

1.1.2 Physical based attack detection

AVs are typically endowed with many sensors, such as laser, radar, camera, Global Positioning System (GPS), and Light Detection and Ranging (LiDAR) [131]. As an AV gleans information from these signals, it adapts its planned path on both small and large scales [15]. These decisions are also guided by communication with other AVs and infrastructure units. However, autonomous systems are vulnerable to physical attacks that manipulate their sensors through spoofing or other adversarial inputs. If the sensor values are incorrect, an autonomous system acting on them can be made to malfunction or may even be controlled to perform an adversary chosen action, making this a critical threat to the success of these systems. Conventional security approaches include those for software security, memory protection, authentication, or cryptography, and are not sufficient to protect cyber-physical systems from attacks originating from the physical world, such as transduction attacks [42]. To address this security gap and protect against these attacks, Physics-Based Attack Detection [44] is receiving increased attention. Specifically, these defenses use a model of physical systems to predict feasible future states or

state-invariants capturing the correlation between the inputs, outputs, and the state of the system. The system can then detect anomalous sensor inputs if their values diverge substantially from the prediction. Physics-Based Attack Detection algorithms include two parts. The first part builds a model of the physical invariants of the system and can be done offline. In the second part, an online tool monitors predicted and observed measurements to see if they fit expectations of the correlations between sensors and the correlations between sensors and actuators.

1.1.3 Anomaly detection

The AVs and CVs technologies [127] rely recently on crash avoidance systems that is based on radars and cameras to detect collision threats by enabling these vehicles to warn their surroundings of collisions and potentially hazardous circumstances. However, as vehicles and infrastructures become more interconnected and automated, the vulnerability of their components to faults and/or deliberate malicious attacks increases. This vulnerability is exacerbated by the increase in vehicle-to-vehicle (V2V) and vehicle-to-infrastructure (V2I) communications. Thus, anomaly detection in the sensor systems enhances safety and reliability of AVs or CVs. Anomaly detection in AVs and CVs sensors is an important but also challenging task. A traveling vehicle could use the most recent history of data to detect anomalies. The presence of an anomaly in the pattern of data collected from the sensor system implies (i) a subset of sensors are faulty, or (ii) there has been a malicious attack. In both cases, it is vital to detect the anomalies and exclude the anomalous data from the decision- making process. An anomaly detection scheme introduces two types of errors; false negatives and false positives. False negative errors can allow falsified data to affect trajectory planning, which could lead to fatal consequences. False positive errors can have consequences that are just as severe (e.g., an unexpected braking from a downstream vehicle), which can lead to an abrupt change in the pattern of observed data. If the vehicle falsely detects such an unexpected change as a fault/attack and discards the information,

it may lead to the vehicle not reacting to such abrupt changes in the network appropriately and in a timely manner, creating dangerous, and potentially fatal scenarios. In order to prevent this type of false positive error, it is necessary for vehicles to incorporate network-level information in their anomaly detection scheme.

1.2 Dissertation Contributions

This dissertation introduces application- level attacks and mitigation which will support building efficient automated and autonomous vehicles with strong security properties.

1.2.1 Automated applications and authentication vulnerabilities

CVs expose a large attack surface since they include many participants and complex functionality. Attacks may target application protocols, networking, and sensing and vehicle control, with the potential to cause accidents, traffic delays, and other harm to the system. The Secure Certificate Management System (SCMS) is a message security standard that has been defined by US DOT; however, it only ensures that cars and road side units have certificates that enable them to participate in communication [45]. In our work, we carry out a security analysis of a previously developed connected vehicle application (Cooperative Adaptive Cruise Control, CACC). We show that even when an attacker does not spoof or modify messages, it does not stop a malicious actor from obtaining a certificate, or a compromised participant with a valid certificate from using it to falsify information in its messages. It is essential to understand the threats faced by CV protocols in order to understand how to design them securely. Keeping this goal in mind, we highlight on vulnerabilities that arise at the application level of CV applications. We present the threat model in Section 5.1. We conduct this analysis in the context of an important CV application called Cooperative Adaptive Cruise Control (CACC). CACC is

used to group nearby cars into a platoon and adaptively control their speed. The vehicles in a platoon are subjected to reduced air drag, as well as improvements in overall traffic flow, driving safety, capacity, and fuel economy. CACC application logic is complex, involving consideration of scenarios such as cars joining and leaving a platoon, merging and splitting of platoons, lane changes, and platoon leaders leaving. These maneuvers are triggered and coordinated through messages. An attacker can exploit this protocol by sending messages with false information leading to a number of possible attacks which reduce the safety and performance of the system. Since CV systems are not deployed and/or generally available for public experimentation, to evaluate these attacks, we use a previously developed implementation of CACC in a state-of- the-art vehicular simulator that is widely used by practitioners and developers (VENTOS [7]. We demonstrate scenarios where vulnerabilities can be exploited to cause safety breakdowns, such as interfering with an emergency vehicle. We evaluate the impact of the attacks using metrics of mobility (traffic throughput) and safety (average separation between cars). We show that attacks can substantially interfere with the operation of CACC, leading to increased vehicular speeds and reduced safety margins. Having established these attacks on the CACC application- level, we need to consider a mitigation framework. We use the classification of the five vulnerability types we introduce to guide the design of the mitigation steps that either eliminate or interfere with them. For example, one class of vulnerabilities occurs when the application logic does not check whether the data in the messages are consistent with other information it has about the system. Some of the message values are impossible to verify due to the lack of independent information to confirm it. Thus, we propose having an alternative source of data (specifically, data from cameras) to validate information in CACC application messages. In our work, we show that the defense indeed mitigates the vulnerabilities we identified in CACC without substantially harming performance.

1.2.2 Physical based attack detection

Achieving autonomy requires processing typically large amounts of data coming from various sensory sources to synthesize situational awareness and guide control decisions towards providing functionality, safety, and performance. Specifically, a fully autonomous vehicle extracts information from the surrounding environment obtained via various sensory devices using different Electronic Control Units (ECUs). As a result, compromising or tampering with any of these components may destabilize an AV or even allow attackers to gain control, causing property damage and bodily injury. An adversary can compromise the AV sensors of a victim autonomous vehicle [33, 95]. For instance, an attacker may use GPS spoofing [62], by leveraging a nearby radio transmitter to create malicious GPS signals leading to erroneous location or velocity estimates. GPS spoofing can be accomplished with inexpensive, commercially available, software-defined radios using open-sourced software. Within our threat model are also *transduction attacks* [44], which inject out-of-band signals to sensors or actuators to cause erroneous measurements [43]. Transducers are components that are responsible for converting physical signals into digital measurements. A transduction attack leverages the physics of a transducer, manipulating physical inputs to try to cause measurement errors to the advantage of the attacker. For example, sound waves can affect accelerometers and make them report incorrect values [121]. Other attacks are shown to be effective, such as using sound to affect gyroscopes [111], lasers to affect camera sensors in drones [35] and LiDaR sensors in cars [31], and intentional electromagnetic interference to manipulate actuators in drones [99]. Some of these attacks have been shown to lead to catastrophic compromise of the system, including drone crashes [111]. In our work, we propose a mitigation system that leverages a number of techniques, including the use of machine learning, to capture the non-linear dynamics of the system, enabling accurate detection of attacks with low false positives, and substantially outperforming previous

defenses. Specifically, our work uses an optimization framework to learn effective Kalman filter (KF) configuration for the characteristics of the vehicle and operating environment. The results of the Kalman filter are improved by applying a genetic algorithm (GA) to tune Kalman filter parameters to obtain well suited parameters before its application to AVs. In order to improve the accuracy of the KF in dynamic environments, our work introduces a new machine learning- based component that estimates the amount of error/residual resulting from the non-linear dynamics of the system, without having to build expensive non-linear models. Finally, to avoid false positives due to noisy sensors and other transients, we leverage a change detection model which analyzes the sensor data to differentiate real changes from transients, further improving accuracy. Our work performs better than previous work in terms of the root-mean-square-error, with negligible false positives. This added accuracy makes is significantly harder for attackers to conduct an attack without triggering the detector, which in turn initiates safe maneuver actions.

1.2.3 Anomaly detection

CV security is critical, since one malicious actor can reach numerous vehicles through connectivity and compromise safety by affecting entire traffic flows. Therefore, it is challenging to ensure that all CVs have proper and timely protections against spoofing attacks [47]. In a CV environment, diversity of the receivers (e.g., vehicles, infrastructures and pedestrians) further increases the system complexity. For example, in the case of the Nissan Leaf, security testers [74] demonstrated how they could gain unauthorized access to control the heated steering wheel, seats, fans and air conditioning remotely. Thus, to fundamentally solve the problem, it is necessary to prevent data spoofing in a timely way. Therefore, we design a new scheme with the consideration of utilizing BlockChain (BC) consensus mechanisms for CV security against cyber spoofing attacks. At its core, our system leverages BC to create a decentralized verification authority, with equivalent functionality to Security Credential Management System.

In particular, we use a data-driven methodology to maintain trusted communications between vehicles in the context of a cooperative ramp merging application, supported using BC. Our results show that we are able to detect malicious vehicles in a quick manner (less than two seconds) using a BC implementation with low computational cost. We believe that our results demonstrate the feasibility and effectiveness of using BC to track trust in a CV environment. We develop a prototype of V2X security mitigation scheme based on BC technology, and a data driven solution for malicious trajectory information. We show how BC consensus can be designed to avoid various spoofing attacks with the help of different vehicular units that act together to validate information coming from any suspicious actor. We perform an extensive simulation to demonstrate efficiency and effectiveness of our scheme using an open-source simulation framework.

1.3 Outline

In Chapter 2, we provide a literature review and discuss the attacks and state-of-the-art mechanisms of secure, connected vehicles' maneuver and autonomous systems. We then present our implemented application -level attacks on Connected Vehicle Protocols in Chapter 3.

In Chapter 4, we present a new defense against application-level attacks on CV systems. This defense models both vehicle level dynamics and inter-vehicle interactions to enable the detection of application-level attacks. Then, we discuss securing autonomous vehicles (i.e., securing a single vehicle from on-board cyber attacks), by Learning Control Invariants and Residual Prediction in Chapter 5.

Moreover, in Chapter 6, we present another proposed mitigation for a secure CV system to ensure safety from falsified information and cyber attacks without any supervising from a central monitoring controller, such as a RSU. This solution is based on developing an algorithm

that uses BlockChain (BC) to maintain trusted communications between vehicles in the context of a cooperative ramp merging application. We finally conclude and present the future work in Chapter 7.

Chapter 2

Related Work

We organize the discussion of related work into five groups: (1) Connected vehicles deployment (2) Attacks against connected vehicles; 3) Attacks Detection in connected vehicles; 4) Attacks against autonomous vehicles; and (5) Attack detection for autonomous vehicles.

2.1 CV deployment

The Intelligent Transportation Systems Joint Program Office (ITS JPO) [5] works with its partners with a fund of nearly 25$ million to support a foundational vehicle cyber security threat assessment for CV applications. Their work includes designing, developing, and operating the security credential management system (SCMS) for the CV Safety Pilot evaluations were conducted in some cities such as Ann Arbor, Michigan, as well as the current US DOT CV pilot studies in NYC [9], Tampa [10], and Wyoming [8]. They developed certification practices to check equipment prior to implementation in the Safety Pilot to ensure that they meet cyber security requirements. The program is also working to ease providing certification services for different industries. The program's primary goal is to improve the best practices for handling foundational electronics control and reliability cyber threat information for existing vehicle fleets.

2.2 Attacks against CVs

Chen et al. [32] performed a security analysis on a system called Intelligent Traffic Signal System (I-SIG). In this system, a real-time vehicle trajectory data is sent through the dedicated short-range communications (DSRC) technology that is acquired by any CV. This data is then used to control the sequence and duration of traffic signals. The system that was attacked includes real deployments at road intersections in some cities such as Anthem, AZ, and Palo Alto, CA. In these deployments, it enhanced the traffic by reducing total vehicle delay by 26.6%. This paper presented an attack that can cause the traffic mobility to be 23.4% worse than that without adopting I-SIG. The attack consists of a packet spoofed to tell the I-SIG of a vehicle approaching from a long distance, causing the traffic light to wait for it, while holding up traffic from other directions. The authors suggested a possible defense that considers scheduling over multiple periods. Amoozadeh et al. [24] performed a security and vulnerabilities risks analysis related to the VANET communication in CV in specific applications including cooperative adaptive cruise control application (CACC). They focused mainly on how to attack a single platoon. The adversary has the ability to falsify, spoof, or replay messages in general to slightly affect the stability of vehicle stream by altering some parameters such as velocity, acceleration, or vehicle location. They considered a CACC vehicle stream that is moving in a straight single-lane highway where all the vehicles use a simple one vehicle look-ahead communication scheme. They did not consider the security credential management system (SCMS) in their simulation. In their work, they examined existing countermeasures, and explored the limitations of these methods and possible ways to lighten negative effects.

2.3 Attacks detection in CVs

[54] developed an architecture where they reduced the conflicts based on a simple comparison of conflicts before and after the operation of their mitigation scheme in this scenario. Recently another study [34] used micro-simulation to analyze the impact of cyber attacks on a platoon of ten CACC vehicles on a single lane. Jamming was identified as the most dangerous cyber attack and resulted in crashes and oscillations in speed. [63] analyzed the cyber risks in the communication medium of an active traffic management system and developed a real-time prototype monitoring system to revert the system back to a safe state of operation under cyber attacks. Researchers in [70], investigated the impact of slight cyber attacks (defined as an attack on a single vehicle for a short interval of time) on the longitudinal safety of CVs. The communicated positions and speeds from preceding vehicles are used as attack factors, and it was observed that a slight cyber attack has a more serious impact on the deceleration period than the acceleration period of nine vehicles. [48] proposed CV Shield, which utilizes the recent advances in hardware-assisted security (e.g., ARM Trust-Zone) to prevent compromised vehicles from sending falsified sensor data. CV Shield can ensure the integrity of the sensor data from their reading to their transmission at the vehicle side. In general, all codes that are related to sensor data reading, processing, encapsulation, and transmission from the rich execution environment (REE), are relocated into the trusted execution environment (TEE). However, manually extracting code sections is laborious and error-prone. Also, the trusted computing base (TCB) size in TEE should be minimized to reduce the attack surface. Thus, they proposed to leverage program slicing to automatically extract code sections and eliminate irrelevant codes in large code bases. The initial results demonstrated that CV Shield could support GPS data reading and eliminate the time overhead of Trust Zone's context switches.

2.4 Attacks against AVs

[123, 79, 60] are based on raising an alert when any electronic control unit is compromised. If an attacker successfully compromises any electronic control unit, a sudden change in the vehicle's state will be used to detect the attack. These schemes mitigate the target system by comparing signals with pre-defined attack patterns known as attack signatures that achieve a low false positive rate. However, it requires maintaining an up-to-date attack database but cannot handle zero-day attacks or even adversary attacks. [104] proposes a hardware-based framework that implements mutual authentication and encryption over the CAN Bus, which adds more overhead to the AV systems and does not prevent sensor or physical invariant attacks. [101] uses Indisputable Code Execution (ICE), a protocol to securely execute codes on a network node from a trusted station based on checking the integrity of the firmware update code and setting up an environment for firmware update. However, by definition, this scheme is not resilient against sensor attacks. Redundancy-based techniques [38, 41] duplicate the essential system components (e.g., controller) and cross-check their states and outputs at run-time for detecting attacks and anomalies. The redundancy can include software and hardware modules. However, this approach requires additional cost and complexity, such as more hardware capacity for multiple versions of the same software.

2.5 Attack detection for AVs

Recently, [33] proposes a framework based on instantiating control invariant parameters to detect physical or sensor attacks without having to reverse engineer the specific control algorithm of a vehicle. However, their algorithm uses a simple linear prediction filter that exploits the geometric properties of the accelerometer and magnetometer, which are well-known nonlinear physical invariants. [95] uses a second-degree nonlinear filter to check statistical differences

between sensor measurements and observed states over time. However, their prediction algorithm uses the first-order Taylor expansion approach to transform the nonlinear system into a linear system which brings some systematic deviations because of ignoring the system's nonlinearity.

Chapter 3

Application level attacks on Connected Vehicle Protocols

3.1 Threat Model

We assume a CV application using Security Credentials Management System (SCMS) [5]. SCMS became available to coincide with the full-scale deployment of devices at three US DOT CV pilot sites (New York, Tampa, and Wyoming) [9, 10, 8]. The current implementation is a proof-of-concept Certificate-Based Authentication system that uses a Public Key Infrastructure [45] for certificate management. Pseudonym Certificates (PCs) are used and rotated to enable message authentication and validation without exposing the privacy of a vehicle by having a permanent certificate. A vehicle can enroll in the system by submitting an enrollment request to US DOT. PC can be obtained by vehicles for a short term, ranging from 5 minutes to few days, and is used for basic safety message (BSM) authentication. On Board Equipment (OBE) uses identification certificates to authenticate itself in V2I applications. However, none of the V2I applications we reviewed require encryption by the OBE at the application level.

SCMS prevents an attacker from falsifying messages from another vehicle as each message gets signed with a certificate. However, SCMS can not prevent a malicious actor from obtaining a certificate and participating in the protocol through replaying the messages while they are valid, or sending its own message, with fabricated data, using its certificate. Although it is currently unclear how well SCMS can function since it is not open source, we assume that it introduces no significant latency. In general, we do not consider message delays, jamming, physical attacks on sensors or controllers, DoS attacks, or any similar attacks to be part of our threat model since our focus is on application level exploitation. It is clear that such attacks are possible, and perhaps can be used in conjunction with application level attacks to amplify their damage. We also do not consider attacks exploiting bugs in the software stack of any of the existing components running on the infrastructure components, or other cars which we consider to be orthogonal to our threat model. We also do not consider physical attacks on the sensors of the vehicles or any sensors deployed by the infrastructure.

In some attacks, we assume that the attacker is a compromised vehicle which uses a radio that is capable of reaching cars farther away than typical vehicular radios and is capable of authenticating itself to the SCMS as a regular vehicle, then applying its attacks in the application level. We assume that the attacker knows the application logic and crafts its actions to manipulate this logic.

3.2 Cooperative Adaptive Cruise Control

In this section, we introduce the Cooperative Adaptive Cruise Control (CACC) application to provide background necessary to understand its potential security vulnerabilities. In CACC, a group of vehicles, with a close spacing between them, can form a platoon if they are traveling in the same direction. Once created, vehicles in the platoon co-operate to travel

at the same speed and make decisions as a group, maintaining reduced clearance gaps between each other, allowing for more efficient use of the highway and reducing the air drag compared to vehicles traveling individually. A Platoon Management Protocol (PMP) controls platoon operations and maneuvers. The leading/front vehicle acts as the coordinator and controls platoon decisions such as the speed, lane changes, and merging with other platoons. Vehicles communicate typically through Dedicated Short Range Communication (DSRC/IEEE 802.11p [46]), although eventually they may use 5G instead [50]. Road Side Units (RSUs) [40] are infrastructure units that are used to coordinate behavior or maneuver across cars, or to maintain shared certain state. Each vehicle has On Board Unit (OBU) that can uses Basic Safety Messages (BSMs) to send some periodic information such as speed and location and receive event messages such as those informing of traffic conditions in an area they are entering.

In our experiments we use PMP, which was proposed and developed by Amoozadeh et al [23]. PMP supports a number of maneuvers representing different operations that platoons could potentially perform. This section introduces some of the primary maneuvers.

Joining a new Platoon (or forming a new platoon): If a vehicle receives a beacon message sent from a vehicle ahead of it, it will evaluate the position, speed, acceleration, and other relevant information to determine whether or not to join the platoon. Beacon messages also contain a Platoon Id, which is a locally distinct number used to distinguished the various platoons in the area.

Split Maneuver: Split maneuver is always initiated by the platoon leader. When the platoon size exceeds the optimal platoon size, the maneuver can be used to break the platoon into two, at a specific position. First, a SPLIT_REQ message is sent to the vehicle where the split should occur. If the request is accepted, a SPLIT_ACCEPT message is sent back to the leader. Subsequently, the leader sends a unicast CHANGE_PL to the potential leader of the new platoon resulting from the split. Finally, the original leader will report split end by sending SPLIT_DONE message.

Merge Maneuver: In this maneuver, two platoons, traveling in the same lane and close to each other, merge to form one platoon. If the leader of the rear platoon discovers another platoon in front of it with capacity to merge, the leader sends a unicast MERGE_REQ to the front platoon leader. Once the front leader accepts the merge request, it sends back a MERGE_ACCEPT message. On receiving this message, the rear platoon leader starts a catch-up maneuver. Upon reaching the front platoon, the rear platoon leader sends CHANGE_PL to all its followers to change the platoon leader to the front leader. Now the followers start listening to the front leader and eventually the rear leader changes its state from leader to follower after sending a MERGE_DONE message.

Leave Maneuver: The departing vehicle initiates the process by sending a LEAVE_REQ message. The leader sends a LEAVE_ACCEPT message and then split process starts. Once the leaving vehicle changes lane, a GAP_CREATED message is broadcast. A merge process begins to reduce the gap until the platoon has the target gap distance between each car.

Change Lane Maneuver: In this maneuver, the platoon leader decides that the platoon needs to change lane. A platoon might need to change lanes if the platoon need to exit the highway or if it has been given instruction from the RSU due to lane congestion. The platoon leader sends CHANGE_LANE instruction to all the other vehicles in the platoon and they perform the maneuver together following the leader's lane change. After that, all the followers send an ACK message to the leader, if they changed the lane successfully.

3.3 Vulnerability Analysis and Classification

It is tempting to consider networked cyber-physical systems such as CV as simply another networked system from the perspective of security, and indeed this is the case with respect to the vulnerability vectors. However, these systems differ in two important aspects with profound implications on vulnerabilities and defenses. The systems are (1) cooperative: they

	Vulnerability	Explanation
V1	Fake message contents	Attacker sends messages with false information
V2	Insufficient information	Critical data not communicated
V3	Inadequate identifier binding	Incorrect binding of physical object to logical object
V4	Incomplete or unsafe protocol logic	Protocol does not consider all scenarios
V5	Trust delegation	Decisions delegated to possibly malicious participant

Table 3.1: Vulnerability Classification in Networked Cyber-physical Systems

coordinate to accomplish a combined outcome; and (2) constrained by physics: protocol logic, as well as misbehavior outcomes are defined with respect to their impact on the system in the physical world, for example, considering both space and time.

The factors, outlined above, lead to vulnerability classes that are tied to the protocol logic and the physical system. Based on our analysis of multiple CV applications, we identified a number of vulnerability classes, which we believe generalize to other networked cyber-physical systems as well. These vulnerabilities arise even if vehicles have a certificate, which, to begin with, is not that difficult to obtain.

The first vulnerability class (**V1**) relies on the ability of the attacker to generate messages with malicious content (e.g., a fake location). By manipulating the information shared to other participants, the protocol logic can be exploited leading to safety or performance compromises. A related class of vulnerability (**V2**) concerns protocols where information that is critical to a sound decision is not considered, perhaps because it is not available, or is not exchanged. For example, the vehicles' lane position and platoon identification number are important parameters that we discovered were not considered when initiating a merge.

A third class of vulnerability (**V3**) relates to ambiguities that arise in *binding identifiers to vehicles*, the act of associating a detected physical information with a moving object such as a vehicle or pedestrian that is known through communication messages. Specifically, sensors can detect physical signals such as proximity to an object and mistakenly associate it with a different

object in the message identifier space. For example, an attacker may pretend to be a platoon leader while a vehicle is attempting to join the platoon, a different vehicle may be mistakenly identified as the attacker/leader.

The next vulnerability class (**V4**) relates to under-specified or incomplete protocol logic. The application logic fails to consider corner cases such as the sudden loss of a platoon leader. In the reference CACC implementation [23], follower cars drive aimlessly if the platoon leader does not communicate with them. Ensuring the robustness of the protocol algorithm is essential for secure application.

The final vulnerability class (**V5**) arises when one object in the system delegates decisions to a malicious or compromised object, thus safety can be compromised. For example, trust is delegated to the platoon leader in CACC which enables arbitrary dangerous maneuvers that can cause crashes and blocking emergency vehicles.

3.4 Application level attacks on CACC

In this section, we present application layer attacks that attempt to exploit the functionality of the PMP implementation of CACC. These attacks were identified from a detailed code review of the PMP implementation. In each attack, we start with explaining the maneuver functionality and consider an attacker that participates in the protocol, sending messages in a way that passes the certificate based authentication and the application logic but results in disrupting the operation of one or more vehicles. We demonstrate the impact of these attacks in later sections.

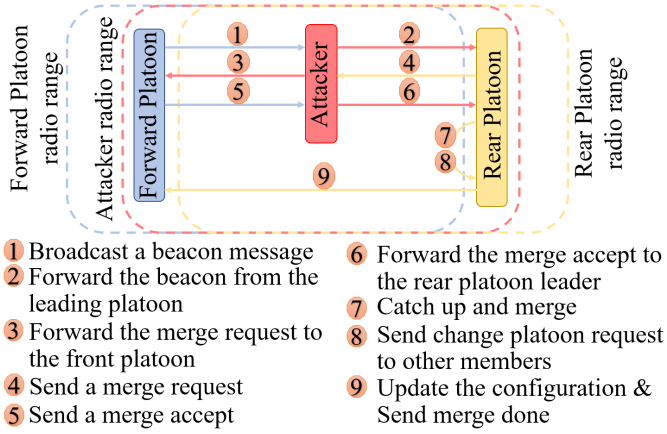


Figure 3.1: Attack scheme of distant merge attack

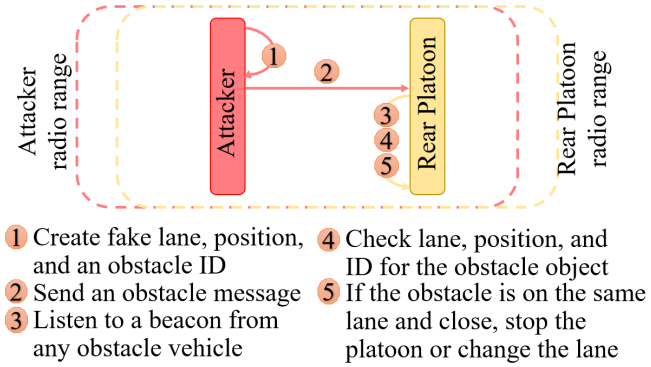


Figure 3.2: Attack scheme of the fake obstacle attack

3.4.1 Attack 1: Merge over large distances

If two platoons are traveling in the same lane and they are close enough while exchanging messages with each other, the PMP application allows them to merge to form one platoon for added efficiency. The application checks prerequisite conditions for the merge, such as, ensuring that the resulting combined platoon does not exceed the size limit. In our experiments, we found out that for two platoons to merge, the rear platoon must receive beacon messages from the front platoon. Then, it measures a certain distance to the last member of the front platoon using its ranging sensor. In our attack scenario, the attacker takes advantage of fake message contents (**V1**) and insufficient information (**V2**) vulnerabilities to target two platoons that are not within the communication range of each other. The attacker in this scenario is located between two

platoons such that it can communicate with both platoons simultaneously and deceive ranging sensor by pretending that it is a member of the front platoon. For a farther distance, the attacker can have a sophisticated radio that can send and receive messages for a longer range.

The attack (Fig. 3.1) begins when the attacker replays the front platoon beacon messages to the rear platoon; since they are merely instantaneous replaying messages, the credentials on these messages are considered valid by the receiving vehicles. Upon receiving these beacons, the leader of the rear platoon will check to see if a platoon exists ahead by using its local sensors to look for a car from the front platoon in the lane ahead, which will be in this case, our malicious vehicle. The rear platoon will then speculate that the front platoon is approaching and initiates merging if the new platoon size is under the predefined permissible threshold (i.e., size of the combined platoon is less than the maximum platoon size).

The rear platoon leader extracts the platoon ID of the front platoon from the beacon and sends a unicast merge request message to the front platoon (which is again relayed by the attacker). The front platoon leader, if it accepts the request, sends a unicast merge accept message, which the attacker then transmits back to the rear platoon. Upon receiving it, the rear platoon leader reduces its time-gap by increasing the speed of the whole platoon to the maximum limit to catch up. At this point, the attack impact shows up when the rear platoon increases its speed for a large distance degrading both safety and economy. Once the inter-platoon spacing becomes small, the rear platoon leader sends change platoon message to all its followers to change the platoon leader to the front platoon leader. Finally, the rear platoon leader sends a merge done message to front platoon leader and changes its state from leader to follower.

3.4.2 Attack 2: Fake Obstacle Attack

A platoon may have automatic incident detection enabled; with this option, the platoon can receive and rapidly react to an obstacle message. Upon encountering an obstacle or accident

in its lane, a vehicle will come to a stop and send an obstacle message with its position to any oncoming vehicles, allowing them to stop or change their lanes when they arrive at the location of the incident. In this scenario, the malicious vehicle exploits the fake content (**V1**) vulnerability and creates a false obstacle message with a specific location in the lane, forcing incoming platoons to slow down until they stop or change lanes. The attack scheme is shown in Fig. 3.2. The fake obstacle attack affects the speed of the platoon and this rapid deceleration can affect safety. The presence of an obstacle is impossible to validate by a distant platoon. Note, that it is possible to combine this attack with *Attack 1* to attempt to create an accident by first speeding up the cars and then forcing them to stop quickly.

3.4.3 Attack 3: Merge across different lanes

In this scenario, we attack two platoons, within the communication range of each other, that are traveling in separate lanes. Critical variables such as lane number and other surroundings information for each vehicle are neither communicated nor checked (**V2** and **V4** vulnerabilities). The attacker can look for a slow platoon in front and try to merge it with a faster platoon from a different lane to slow down traffic flow.

The attack (Fig. 3.3) starts when the malicious vehicle is in front of the rear platoon, and sends messages pretending to be a part of the other platoon (in another lane). This can be done by manipulating the platoon ID parameter in Basic safety message. The rear platoon will see the attacker vehicle using its LiDAR sensor and assumes that the attacker is part of the platoon (**V3**). Information such as Lane ID is neither communicate nor checked. It then begins a merging maneuver. As consequence, the adjacent leading platoon leader sends a merge accept message. As a result, the rear platoon leader increases the speed of the platoon to catch up. Afterwards, the attacker leaves its location and the rear platoon leader sends change platoon to all its followers.

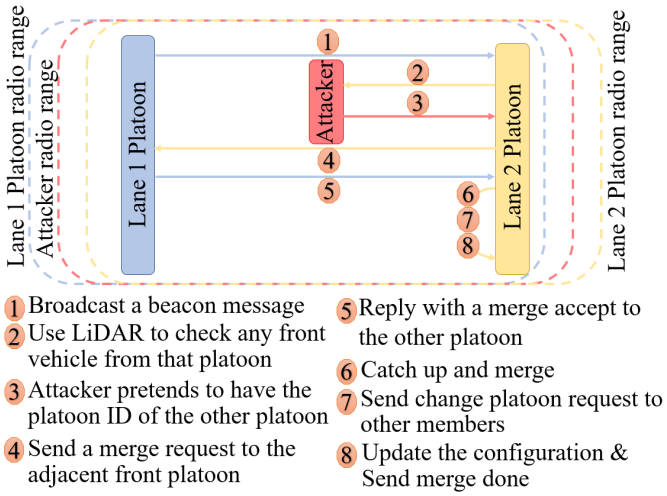


Figure 3.3: Attack scheme of merging across lanes attack

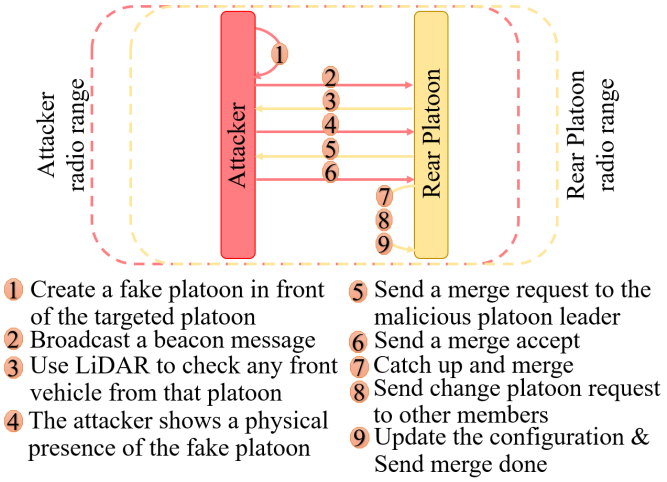


Figure 3.4: Attack scheme of platoon takeover attack

3.4.4 Attack 4: Platoon Takeover

This attack is conceptually similar to the *Attack 3* except that there is only one platoon (the rear platoon), with the attacker attempting to become its leader. The attacker counts on different vulnerabilities but mainly on the fake message contents (**V1**) vulnerability by pretending to be the leader of the fictitious front platoon by generating any logically consistent description of the front platoon such as the locations and speeds of a fake platoons' members in front of the victim platoon. The attacker transmits the fake messages for each false vehicle of the fake platoon. The rear platoon leader will notice the attacker through the LiDAR sensor and initiate

a merging maneuver since it believes that this is the platoon in front of it that it listens to. The attacker responds to all requests from the rear platoon. This leads to the completion of the merging process. The platoon is now under the attackers' control and can be manipulated in a dangerous manner as we show in Section 3.5.1, exploiting the trust delegation (**V5**) vulnerability. We show the steps of this attack are shown in Fig. 3.4.

3.5 Experimental Attack Scenarios and Results

In this section, we first describe the simulation set up used in the experiments. We then present an experimental evaluation of the proposed attacks and evaluate their impact on the traffic system with respect to safety and performance. Given the limited availability of deployed CV applications, and the closed nature of these systems, we elected to evaluate the attacks using simulation. We used VENTOS (VEhicular NeTwork Open Simulator), an extension of Veins [109]. Veins integrates a C++ simulator for studying vehicular traffic flows, collaborative driving, and interactions between vehicles and infrastructure with another simulator which models communication through a DSRC-enabled wireless communication. Veins combines two widely used simulators, Simulation of cars/physics simulator (SUMO) [4] and OMNET++ [2]. SUMO is an open-source road traffic simulator developed by the Institute of Transportation Systems at the German Aerospace Center and serves as the traffic flows physics simulator. This framework has been used in hundreds of studies from academia, industry, and the government (a partial list can be found on the project [6]). VEINS uses SUMO's Traffic Control Interface, TraCI, to communicate simulation commands to it. OMNET++ is an open-source simulation package and carries out the wireless communication simulation. We configure it to use the models for the IEEE 802.11p [46] protocol, a standard adopted for V2V communication. VEINS has been used by other researchers to simulate connected vehicle applications [98], [56]. We use Wave Short

Message Protocol (WSMP) to carry beacon and micro-command messages. These messages are directly sent to the data-link layer which uses continuous channel access based on IEEE 1609.4. The channel frequency is 5.89 GHz with a data rate of 18 Mbps and transmission power of 10 mW.

3.5.1 Dangerous Attack Demonstrations

First, we demonstrate the potential impacts of the attacks using two specific scenarios, one causing a collision and the second interfering with and delaying an emergency vehicle.

Causing a Collision: In this attack, the followers of a platoon that is controlled by a compromised leader, fail to see and stop for stationary or slower vehicles. The malicious car may have acquired leadership of the platoon using the platoon takeover attack. The attacker can suddenly veer out of a lane without informing the followers to slow down or change lanes. The followers' braking systems may not be able to stop if an obstacle appears immediately in their path. We can see the sudden stop then collision at the time 60s for the victim vehicles in Fig. 3.5. After investigating this scenario in detail, we discovered that vehicles in the platoon were not keeping a safe distance between each other. Instead, they were delegating trust (**V5**) to the platoon leader (the attacker), trusting that the leader will maintain safe separation from any obstacles.

Emergency Vehicle Interference: We again start with the attacker using the *Platoon Takeover*

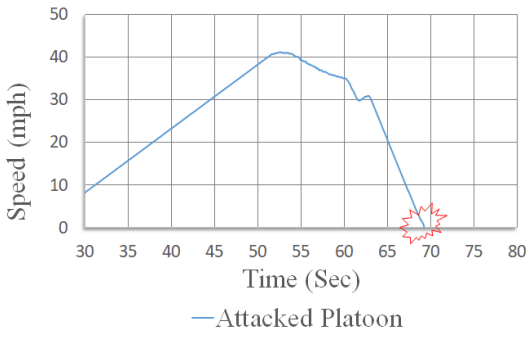


Figure 3.5: Speed profile using a collision attack

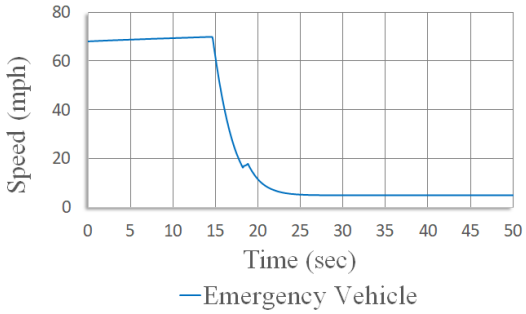


Figure 3.6: Speed profile using an emergency vehicle attack

attack. The attacker slows down the whole platoon then makes some followers move to another lane. If an emergency vehicle (police or ambulance) is coming fast in that lane, a slow vehicle on the same lane will make it much slower or even stop it, as shown in Fig. 3.6. This can cause catastrophic slowdowns in real life (e.g., potential loss of life). Other approaches to delay an emergency vehicle can be devised, for example, using the merge across different lanes attack.

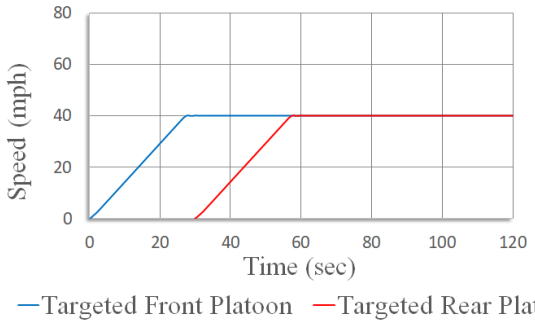


Figure 3.7: Speed profile without using a distant merge attack

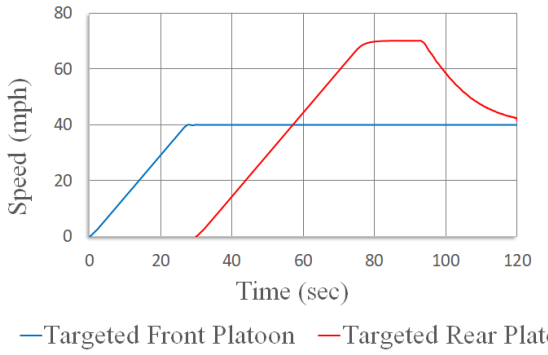


Figure 3.8: Speed profile using a distant merge attack

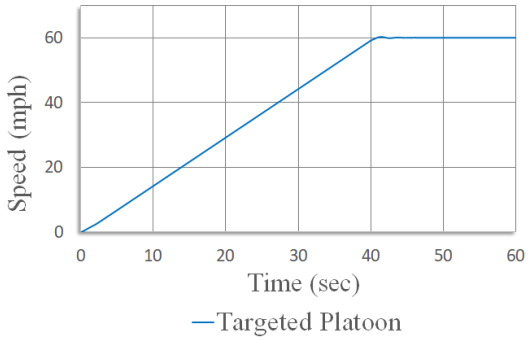


Figure 3.9: Speed profile without using a fake obstacle attack

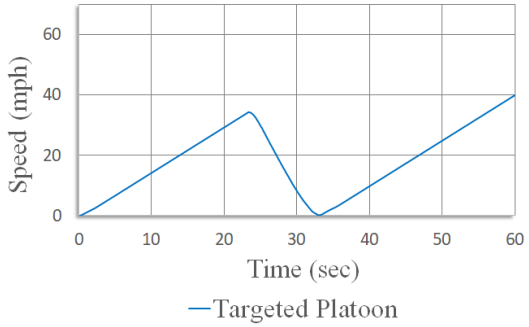


Figure 3.10: Speed profile using a fake obstacle attack

3.5.2 Isolated Attack Scenarios

In this set of experiments, we investigate vehicle performance after implementing the four different attacks described in Section 3.4 isolating the impact on just one or two targeted platoons. These scenarios allow us to evaluate the isolated impact of the attacks.

Attack 1– Distant Merging attack: Our intention in this attack is to make some platoons go to the catch-up process where they speed up abnormally for some time potentially degrading both safety and efficiency. Fig. 3.7 shows the average speed of two platoons in the scenario in the absence of an attack. The rear platoon starts a little later, but both platoons accelerate to 40mph before cruising at that speed. Fig. 3.8 shows the behavior of the platoons in the presence of the attack. In this case, the rear platoon accelerates aggressively, reaching the maximum velocity, in an effort to catch up with the front platoon.

Attack 2– Fake Obstacle attack: From Fig. 3.9, we see a platoon of 3 cars accelerating to 60mph. After initiating the attack starting around time 20s, we can notice how the platoon suddenly comes to a halt as shown in Fig. 3.10. This occurs for a certain time then the platoon changes the lane and accelerates again, but the attack can be repeated.

Attack 3– Merging platoons across lanes: In this scenario, two platoons travel on different lanes where the front platoon is slower than the rear one. The attacker realizes that both platoons are close to each other and locates itself in front of the rear platoon. Next, the attacker initiates the merge maneuver as described in *Attack 3*. When the attack succeeds, all the members of the rear platoon will follow the front platoon (despite being in a different lane) and travel according to its speed as shown in Fig. **??**. In this case, the lower speed platoon slows down the traffic flow. In another case, the rear platoon may be tricked to go faster than the optimal speed for the lane, compromising safety.

Attack 4– Platoon Takeover Attack: The attacker starts with sending different beacon messages

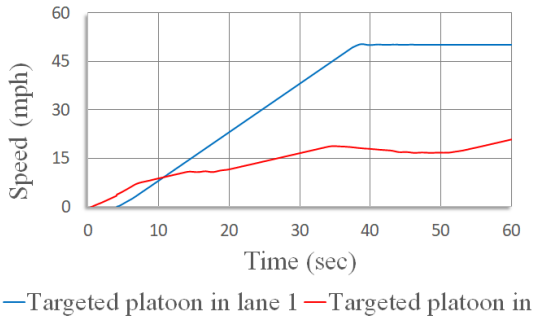


Figure 3.11: Speed profile without using a merging across lanes attack

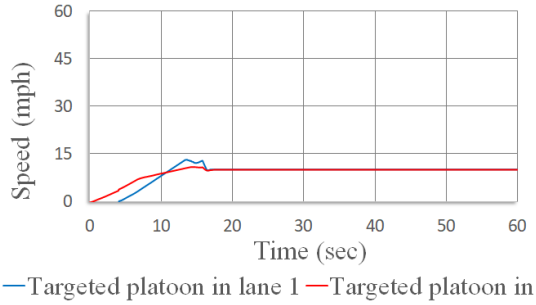


Figure 3.12: Speed profile using a merging across lanes attack

pretending that they come from a front platoon. Once the platoon finds that the leading vehicle on the same lane is the last platoon member that it listens to (through its LiDAR sensor), it will then start the merging process. After the merging succeeds, the attacker now acts as a platoon leader and controls this platoon in any way it desires within the platoon operational parameters. For this example attack, the attacker decreases the platoon velocity and then repeatedly changes the lane of the platoon in order to affect as many lanes as many as possible. Fig. **??** shows the platoon speed changes.

3.5.3 Attacks within traffic scenarios

Next, we evaluate the impact of the attacks when applied as part of an active traffic scenario. We use different metrics to quantitatively analyze the effects of the attacks on *Mobility* and *Safety*. For mobility, we use two metrics: (1) **Average speed** of vehicles is a common metric

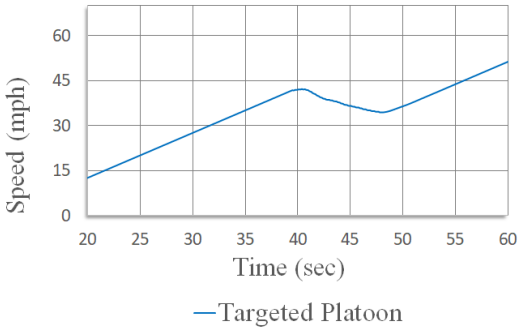


Figure 3.13: Speed profile without using a platoon takeover attack

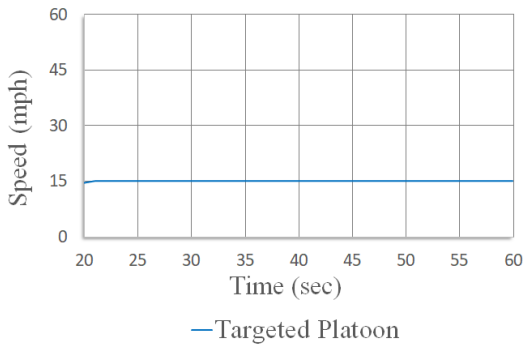


Figure 3.14: Speed profile using a platoon takeover attack

for mobility; and (2) **Flow** of traffic, is defined as the number of vehicles passing a point on the road in a given time. To measure safety, we also use two metrics: (1) **Average speed difference** between consecutive vehicles measures the differences in speed among vehicles. This metric is known to correlate with the onset of collisions and near-collisions; and (2) **Time-to-Collision (TTC)** [78] is metric for safety which measures the time taken for a vehicle to collide with the vehicle in front of it, should they maintain the same speed. TTC of vehicle i at instant t can be calculated as follows,

$$TTC_i(t) = \frac{D_i(t) - D_{i-1}(t) - l_i}{V_i(t) - V_{i-1}(t)}$$

here, $V_i(t)$ stands for the speed of the vehicle i at instant t ,l_i is the length of the vehicle i, and $D_i(t)$ stands for the location of the vehicle i.

California Department of Transport provides real time traffic condition through Performance Measurement System [3] by using various sensors installed in the state's most highways sections. We use data from a section of the highway I-5 in south California and generate scenarios with vehicles entering stochastically following the observed distribution. Each scenario, ran for 5 minutes, simulates the entrance of traffic into a highway section of length 6 miles. We assume that all vehicles are CV enabled to avoid making assumptions on the interactions of CV and non-CV vehicles. We configure about 25% of the vehicles to form platoons of different sizes. The maximum speed for the road is 70 mph. The communication range for each vehicle is 300 meters. Each road has five lanes and approximately evenly spaced road side units (RSU) such that all points in the highway are in range with at least one RSU.

Attack 1–Distant Merging Attack: Fig. 3.15 shows the effects of attack 1 on the average speed, flow, average speed difference, and average TTC for the scenario. The attack causes an increase in average speed and flow of traffic. Even though the flow of vehicle increases by a small amount, the attack causes vehicles under attack to travel at a much higher speed, thus compromising

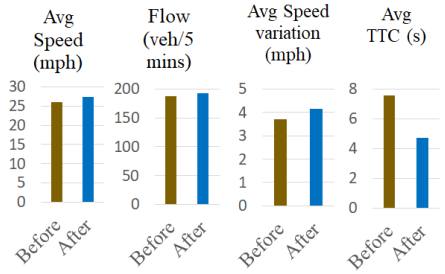


Figure 3.15: The impact of the distant merge attack on the traffic flow

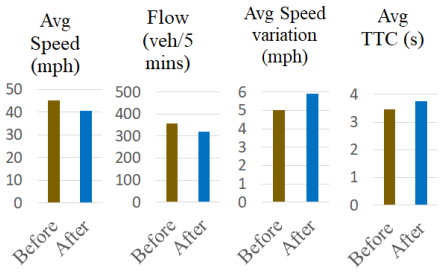


Figure 3.16: The impact of the fake obstacle attack on the traffic flow

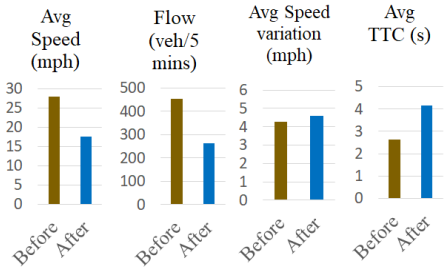


Figure 3.17: The impact of the merge across lanes attack on the traffic flow

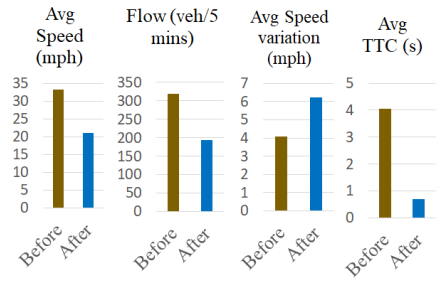


Figure 3.18: The impact of the platoon takeover attack on the traffic flow

safety, which is reflected by the increased average speed difference and reduced TTC. Even though the flow of vehicle increases by a small amount, distant merge attack causes vehicles under attack to travel at a much higher speed, thus compromising safety.

Attack 2– Fake Obstacle Attack: Fake obstacle attack causes the traffic to slow down potentially abruptly, similar to the slow down due to road site construction. Thus, it has slight adverse effect on safety, with increased average speed difference and TTC, but a large effect on the mobility, with decreased average speed and flow, as depicted in Fig. 3.16.

Attack 3– Merging across lanes: In this attack, the attacker connects the flow of traffic of two or more lanes, forcing a faster platoon to slow down. The effect of the attack is shown in Fig. 3.17. Average speed difference increases only slightly, while TTC increases, leading to a marginal impact on safety. However, the flow of the traffic is severely hindered which is shown by the steep drop in average speed and flow.

Attack 4– Platoon takeover: In this attack, the attacker takes over the control of a platoon and can control it fully. This is the most dangerous form of attack that the attacker can carry out. Although different arbitrary maneuvers are possible once the attacker controls the platoon, we went with a speed reduction and repeated lane change maneuvers. Both safety and mobility metrics are highly affected by this attack, as seen in Fig. 3.18.

3.6 Potential Mitigation

Our eventual goal is to develop a defense approach that is automated and can mitigate the vulnerability classes we identified in Table 3.1, thus making the protocol logic more secure in a principled way. The general defense approach relies on augmenting the information available to vehicles with a redundant source of information that enables detection of incorrect or malicious information, and makes the protocol logic more robust. If such a source of redundant information is available, the veracity of the exchanged messages can be checked before conducting critical actions within a maneuver, thus addressing **V1** and **V2** vulnerabilities. To give an example, if a merge is attempted with a far-away platoon, the requested platoon should check if the distance

of the front platoon is within the merge range; previously, this was assumed from the fact that the messages were received from the front platoon, an assumption that can be exploited by an attacker that replays a message (effectively extending its reach) or to use higher power radio to increase its range. Moreover, this defense substantially reduces the opportunities for **V3** attacks since it becomes more difficult to create wrong bindings between message sources and other physical objects. This check would defeat the replay attack that allows the adversary to initiate a merge. **V4** can be addressed by in depth protocol testing and analysis. Finally, **V5** can be addressed by either avoiding trust delegation or verifying delegated decisions.

We collect complementary information through a reliable sensory system to protect against fake message contents (**V1**). Validating protocol components by linking message contents and redundant sensor data is also desirable for a reliable decision. The consistency of the application and environment constraints using a robust algorithm need to be considered to prevent a message with clearly unfeasible information to be acted on and ensure that the resulting action is consistent with the protocol logic. If everything checks out, a final decision will be assigned to protocol controller to lead the required action.

3.6.1 Preliminaries and Assumptions

RSU: Defense components infrastructure: The main component that we rely on in our scheme is the road side unit (RSU), where its hardware and software components are specified by US DOT [89]. The RSU is a more sophisticated and more protected component of the system deployed and managed by the infrastructure provider, making it an attractive component to root defenses. It is expected to operate unattended in harsh outdoor environments for extended periods of time (typical Mean Time Between Failures of 100,000 hours). It detects and auto-recovers from minor software failures, transient power spikes, and power interruptions.

Moreover, we consider a case where RSUs are reachable from any point on the highway as a proof of concept, but the protocol can be made to act conservatively in Safe mode when RSU are not reachable.

We note that without relying on the RSU, the alternative is to reach consensus between the different cars which is an interesting possibility. A naive implementation could be too costly to achieve on-demand, and therefore we elected to root our defenses in the RSU.

Safe mode and functionality of RSU: We identify a safe operation mode for platoons with respect to any maneuver or protocol state. The goal of the safe mode is to be used as a cautious behavior when protocol exchanges are in progress, or when a decision cannot be made. For example, the platoons could either maintain their speed or slow down and wait for confirmation after sending a maneuver request. The defense proceeds by having the RSU check the the proposed action against the configuration of the platoon (e.g., the location of each member of the relevant platoons from all basic safety messages (BSMs) it collects). The RSU uses, as a source of redundant information, a video tracking system to track the vehicle locations. The system also maps any incoming messages to vehicles based on the geographic information to check the consistency of messages being sent by any particular vehicle. Other sources of redundancy are also possible, for example, exchange of past information from nearby RSUs for vehicle tracking, or alternative real time sensors. Our proposed video tracking system is feasible: many vehicles tracking systems using video cameras have been proposed [132], [130]. We would next see how the defense would work for the previous attacks.

3.6.2 Defense overview

Defense against Merging attacks: For *Attacks 1, 3*, and *4*, the defense starts by allowing the back platoon to send a merging request to RSU. After receiving the maneuver request, the RSU verifies the relevant information. Then, it tests if the merge process is applicable

or not by inspecting the constraints between the platoons such as making sure that the distance between them is within the permissible range. If all checks pass, an approval reply is sent to the two platoons to start merging. If the maneuver confirmation is received and leader, for any reason does not exist, the platoon members can start a voting process where they study the collected BSMs and check its neighbor vehicles through its sensors to choose their leader to control the maneuver.

Defense against Obstacle attacks: For *Attacks 2*, RSU carries out the same steps regarding requesting a maneuver. For this scenario, it checks specifically if the obstacle and the incoming platoon are in the same lane or not and, if yes, the distance between them. Then, the RSU will send an approval reply to stop the coming platoon or change its lane. In the meantime, the traveling platoon leader will go to the safe mode where it moves within the safety speed limit which we defined here to be below 20 mph. Generally, it is sufficient to ensure the ability to stop in case the obstacle message is confirmed. If the platoon does not receive any confirmation for the obstacle maneuver until the obstacle location, it can start the backup protocol where it can stop or change lane. Fig. 3.19 shows the general protocol for the RSU. Algorithm 1 shows the steps for the platoon leader.

3.6.3 Evaluation

We implemented the defense logic within the simulator. We emulate the video tracking by using the ground truth value of the location and adding Gaussian noise to it with a mean of 2 meters. We augmented the application with the defense by following the mitigation steps discussed above. Fig. 3.20 demonstrates that with the defense in place, the attack impact is mitigated from all attacks other than the fake obstacle attack where it has a minor effect. The effect is due to the delay in confirmation from the RSU, during which the safe mode reduces performance whether there is a real obstacle or not. This mitigation's approach will also be able

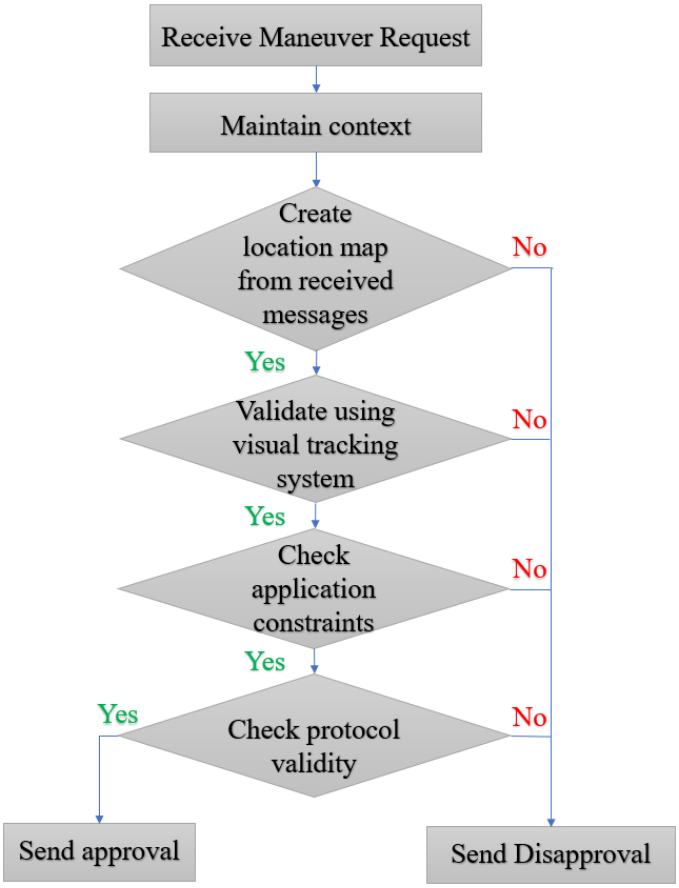


Figure 3.19: Pre-Maneuver Protocol process for RSU

Algorithm 1 Pre-Approval protocol for Platoon Leader

```
1:  procedure PRE-APPROVALPROTOCOL
2:      SendManeuverRequestToRSU()
3:      Change to SAFE mode                          ▷ Wait for RSU response
4:      loop:
5:      if Disapproval Received then return AbortManeuver()
6:      end if
7:      if Approval Received then return StartManeuver()
8:      end if
9:      if Time Out Exceeded then return Exit-loop
10:     end if
11:     goto loop                                    ▷ Time Out NOT Exceeded
12:     StartBackupProcedure()
13: end procedure
```

to stop the dangerous attacks described in Section 3.5.1. This is due to the fact that those attacks are bases on the basic attacks demonstrated in Section 3.4 but used in specific scenarios.

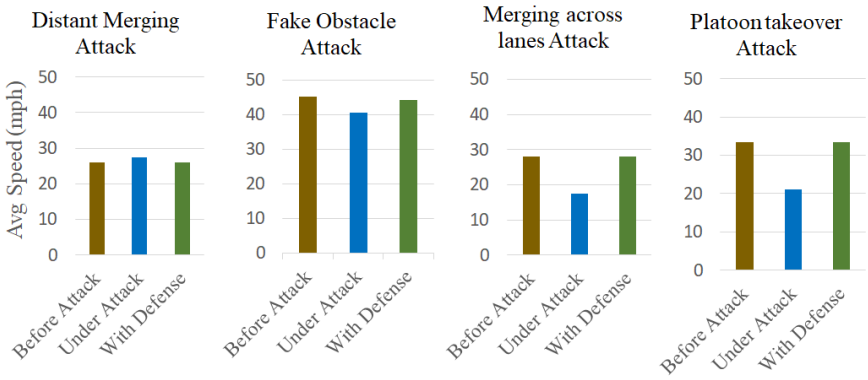


Figure 3.20: The effect of the potential defense on the studied attacks

3.6.4 Discussion

A concern with any defense strategy that requires additional operations is delays in making decisions, while information is validated. However, we believe that the redundant information can be prepared proactively so that the check is often local. Moreover, it is critical to deploy the safe backup operation while decisions are being taken in any cyber-physical system, prioritizing safety over performance.

The approach heavily relies on the visual tracking system and sensors for more reliable decision, which may not be available in all vehicles and in some areas. Thus, we accept that it is a strong assumption on our part to assume that such redundant data will always be available to the decision making system. We can see from Fig. 3.20 that safe mode does not significantly degrade performance in CACC application. Nevertheless, we will carry out analysis and performance measurements on other CV applications to justify this statement in the future. In the future work, robust algorithms may be employed to detect all the different attacks in the early stages.

3.7 Vulnerabilities in other protocols

In this section, we analyze other protocols and classify the vulnerabilities using the attack vectors defined in Table 3.1. We performed code reviews of two protocols available on the US DOT open source CV protocol repository [11]: (1) Intelligent Intersection Management and (2) Eco-Traffic Signal Timing.

Intelligent Intersection Management has shown great potential in improving transportation efficacy especially for autonomous vehicles where it connects with them wirelessly and schedules their intersection crossing steps. In [32], they proposed to use existing infrastructure-side sensors to stop malicious messages. For example, vehicle detectors buried underneath the stop bar of each lane can be used to measure aggregated traffic information. After analyzing the scheme, we found out that malicious messages can still be sent to manipulate the application and increase total delay time. This is due to inadequate identifier binding (**V3**) vulnerability, where sensors do not correlate the messages with the vehicles and do not give the exact location for each vehicle.

Eco-Traffic Signal Timing application aims to improve traffic signals delays thus reducing environmental impact. It processes real-time and historical CV data at signalized intersections to reduce fuel consumption and overall emissions. In this application, we discovered that vehicle trajectory data can be subjected to fake message contents (**V1**) and inadequate identifier binding (**V3**) vulnerabilities. We were able to implement exploits to manipulate the timing phase for any lane based on sending malicious vehicles information.For both applications, the defense principles we introduced can be adapted to mitigate these vulnerabilities.

Chapter 4

Securing Autonomous Vehicles by Learning Control Invariants and Residual Prediction

4.1 Sensor Modalities and Threat Model

Our goal is to defend against attacks that target sensors either directly spoofing inputs to them or indirectly manipulating them using transduction attacks. Sensors are critical to AV operation since they are used to create situational awareness necessary to plan and execute actions correctly, efficiently, and securely. In this section, we provide some background on common sensor modalities in AVs, as well as their use in constructing state space estimation. We then present the threat model we assume in this paper.

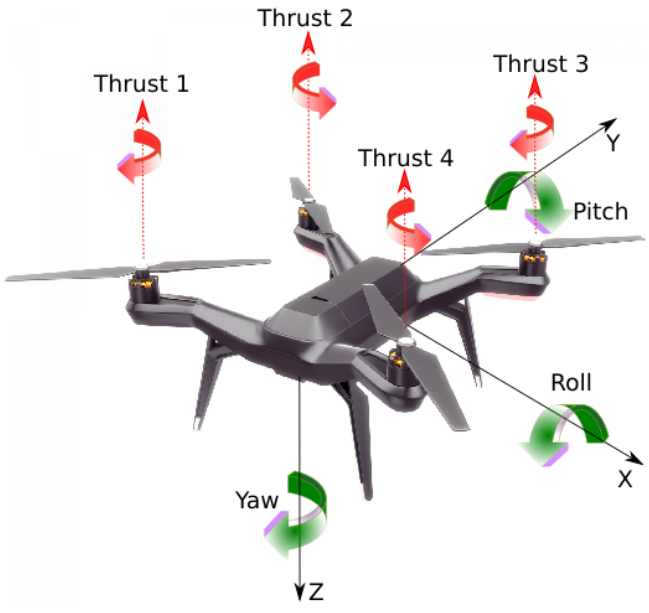
AVs [116] typically include a mix of sensors such as radar, camera, ultrasonic, Light Detection and Ranging (LiDaR), Inertial Measurement Units (IMUs), and etc. Radar sensors monitor the position of neighbouring vehicles. Video cameras can be used to detect traffic

lights, road signs, as well as track other vehicles. Ultrasonic sensors detect curbs and other vehicles during parking maneuvers. LiDaR sensors can carry out ranging, detect road edges, and identify lane markings. IMUs [36] detect the motion of the vehicle, combining a 3-axis linear accelerometer and 3-axis gyroscope to track a vehicle within six axes of motion. Specifically, IMU tracks both linear (X,Y, and Z) and rotational components: (1) pitch, rotating a vehicle upwards or down-wards; (2) roll: rotating, the vehicle sideways; and (3) yaw: rotating, the orientation of the vehicle. These six axes allow the full vehicle position and orientation to be tracked in real-time.

These sensor streams are processed by software modules which use them to generate and adapt trajectory paths, which are effectuated by sending control signals to the vehicle's actuators to control acceleration, braking, and steering. To visualize these axes, the inertial frames of a quadcopter and a car are shown in Fig. 4.1 and Fig. 4.2 respectively.

Threat Model: We consider attacks where one or more of the sensors on an AV are interfered with by an attacker either directly or indirectly (e.g., using transduction attacks) [92]. Since these sensors are critical to the AV's estimation of its own behavior and that of the environment,

Figure 4.1: Quadcopter motion axes and thrust controls



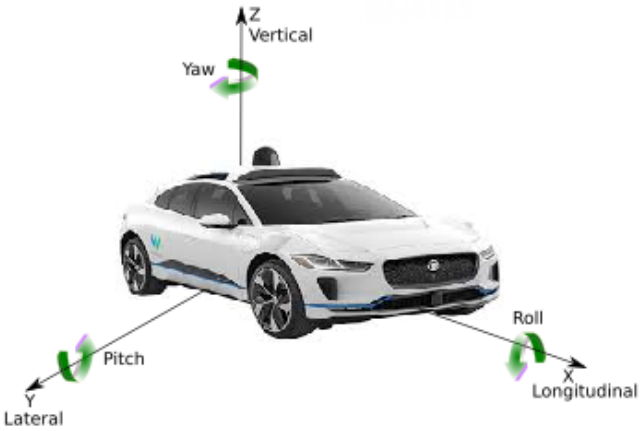


Figure 4.2: Car motion axes

compromising sensors may lead to erroneous estimates of its operating state, leading to control actions from the AV that serve an attacker's goal. For example, the attacker may control the vehicle to cause property damage, block emergency traffic, or cause accidents and bodily injury.

Prior work has shown that a range of common sensors are vulnerable to attacks including those that target IMUs [121, 111, 122], RADAR sensors [72], LiDAR [31, 91, 102], ultrasonic sensors [72], camera sensors [35, 72, 91], and GPS signals [52, 120, 84, 133]. The weaknesses enabling these attacks are specific to the different sensor modalities and their implementations. For example, GPS signals [92] do not contain authentication information and are susceptible to spoofing attacks. Conversely, LiDAR [31] is used for measuring distances to surrounding obstacles using infrared lasers, can provide 360° viewing angles and generate 3-dimensional representations of the road environment. A LiDAR spoofing attack can be performed by replaying the LiDAR laser pulses from a different position to create fake points further than the location of the spoofer [91].

Our threat model is similar to the threat model in previous research efforts so that an adversary can inject false signals in one or more of the sensors used by AVs at a time. Our defense can also be used to predict and monitor actuator commands that are potentially controllable by an attacker. We do not consider a powerful attacker that has complex hardware base and can inject

signals to all sensors and actuators simultaneously to mimic specific physical events or situation. An attacker may gain access in different ways with different capabilities [55] either by physical access to the system, or remotely through open interfaces or through remote interference with the sensors.

We assume that the attacker cannot compromise or bypass our invariant-checking module. We also assume that the attacker does know the physical properties of the vehicle (e.g., weight or frame shape), or low-level control parameter settings. The main strategy of these attacks is to inject a time series of biased attack values so that $y^a = y + bias$, where y^a is selected to harm the system. We also consider sophisticated attacks, such as those where the attacker relies on a machine learning model, and it aims to generate adversarial example where y^a is selected so that it causes incremental drift in the state of the AV without sudden changes to avoid detection by anomaly detection algorithms.

4.2 System invariants and their use in defenses

We next provide a brief review of the autonomous system's dynamics, including aerial and ground vehicles, which serve as the basis of the models used within AVMon. We then show how these models are used as part of a physics-based attack detection defense.

4.2.1 Physics-based System invariants

Aerial vehicles, such as quadcopters [26], operate by controlling variable torques and thrusts through four rotors instead of the two used in standard helicopters. The motors in the quadcopter are arranged in pairs along the horizontal and vertical axes, with the forward pair rotating clockwise and the horizontal pair rotating counter-clockwise. This design produces reaction torques from the pairs of motors that oppose each other, providing equilibrium if they are all operating at the same speed. The elimination of the rotating moment allows the vehicle

to maintain a constant heading while hovering. Yaw is controlled by varying the motors pairs to create a non-zero net counter torque. Altitude is managed by changing the thrust from each motor by equal amounts to provide a net thrust vector and without a rotational moment. To move laterally, the relative speed of each pair of motors varies to create the desired lateral thrust offset.

A quadcopter can move in six degrees of freedom; longitudinally (forward and backward), vertically (upward and downward), and laterally (right and left), by controlling the differential thrusts to the rotors. It can also move rotationally among each axis to produce roll, pitch, and yaw movements. The basic quadrotor parameters that depict Euler angles [58] including roll, pitch, yaw, and body coordinate frame, can be shown in Fig. 4.1. The dynamical model of a four-wheel vehicle is also well studied [19, 88]. This model has two degrees of freedom that are represented by the vehicle's lateral position and the vehicle yaw angle. The vehicle's lateral position is measured along the lateral axis of the vehicle to the vehicle's center of rotation. The vehicle yaw angle is defined as the angle between the vehicle's longitudinal axis and an axis parallel to the surface of the earth in the earth-fixed coordinate system.

4.2.2 Physics-based attack detection

Monitoring the physics of cyber-physical systems [97] to capture sensor attacks is a growing area of research. Our contribution is to substantially improve these predictions for AVs by improving the KF estimator by learning its optimal configuration, leveraging residual learning to compensate for nonlinear dynamics, and using context information to filter out transients and reduce false positives. Physics-based attack detection can be thought of as a security monitoring system that creates a time-series prediction model of sensor readings for the autonomous system and identifies anomalies as deviations between the predicted and actual sensor readings. Thus, such a framework consists of 1) Physical model prediction and 2) Anomaly detection. Physical model captures the relationship between the control inputs and the system dynamics. For example,

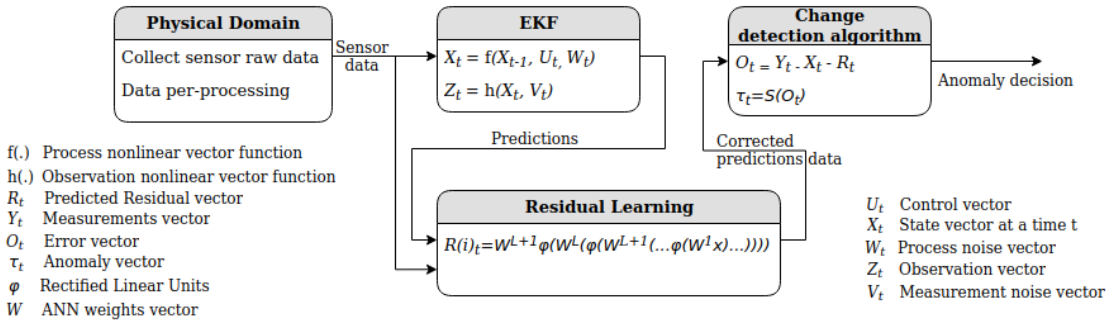


Figure 4.3: AVMon design overview for the online invariants monitoring

to obtain the following time frame position and velocity values for a ground vehicle, matrix A has to be defined as shown in Equation 4.1 which reflects the dynamical model of the ground vehicle. Based on Equation 4.1, the position at any time is the position at the last time instant updated with the velocity multiplied by the time step.

$$\begin{bmatrix} x_{t+1} \\ y_{t+1} \\ \dot{x_t}_{+1} \\ \dot{y_t}_{+1} \end{bmatrix} = A * \begin{bmatrix} x_t \\ y_t \\ \dot{x_t} \\ \dot{y_t} \end{bmatrix} \Longrightarrow A = \begin{bmatrix} 1 & 0 & \Delta t & 0 \\ 0 & 1 & 0 & \Delta t \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{4.1}$$

A physical vehicular system can predict the expected future measurements using a state-space representation that describes the physical system as a set of inputs, outputs, and state variables. In general, the control invariants model can be represented as follows [73]:

$$\begin{aligned} x(t+1) &= Ax(t) + Bu(t) \\ y(t) &= Cx(t) + Du(t) \end{aligned} \tag{4.2}$$

where $x(t)$ is the state variables, $u(t)$ is the system inputs and $y(t)$ is the system outputs. Equations 4.2 determine the next state and output of the system based on the current state and control signals. Specifically, A, B, C, and D are matrices modeling the state and inputs of the system as

follows: A represents the time-invariant dynamic state matrix; B, the time-invariant input matrix; C, the time-invariant measurement matrix; and D, the time-invariant feedforward matrix.

4.3 AVMon Design overview

Fig. 4.3 shows AVMon design. It consists mainly of sensors pre-processing, prediction process using KF, residual learning, and anomaly detection function. These components work together in a real-time/online manner to achieve substantially higher attack detection accuracy by improving the physics-based prediction and anomaly detection components. Generally, three types of KFs can be used for physical states estimation. In our work, we use the Extended Kalman Filter (EKF) that was designed for nonlinear system estimation and filtration. AVMon starts by receiving the sensor data as input and uses it to predict the next state in the prediction model. Then, the prediction data is compared to the sensor data to carry out anomaly detection, as shown on the right-hand side of Fig. 4.3.

AVMon improves the prediction process using two ideas; First, it uses an optimization algorithm that is executed offline to configure the primary EKF module to operate more accurately with respect to the AV parameters. Second, it also carries out residual learning to compensate for the nonlinear dynamics of the model that are not captured effectively by the EKF. Finally, in the anomaly detection process, a time series of residuals r_k, i.e., the difference between the received sensor measurement y_k and the predicted or expected measurement $\hat{y_k}$, is used to detect unusual deviations and raise an alarm if the sensor values are sufficiently different from the predicted values. We improve anomaly detection by using a change-aware model instead of just looking for deviations between predictions and sensor data to monitor the sensor data for self-consistency over time and reduce false positives.

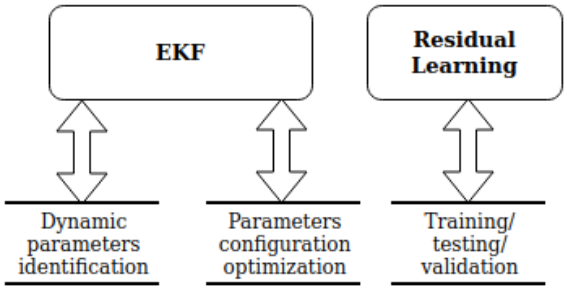


Figure 4.4: AVMon offline learning and optimizing

Fig. 4.4 shows the total offline or training components required for each subsystem within the design. In this section, we describe each of these components in detail.

Preliminaries: Data Collection and Preprocessing

We collect the vehicle's operation profile data, including the series of input states (e.g., velocity and acceleration) and inputs (e.g., latitudes and longitudes) values from the sensors. At the same time, we also pre-process the data to convert it to the variables needed in the physical model. For example, we obtain the GPS readings that correspond to the latitude, longitude, and altitude to convert them to the corresponding flat-Earth coordinates (X, Y, and Z). Then, we use these coordinates to estimate the position of the AV object (e.g., aerial or ground vehicle) in meters. Another example is using the Inertial Measurement Unit (IMU) sensor data that the AV employs by receiving the 3-axis accelerometer, 3-axis gyroscope, and 3-axis magnetometer variables to calculate the orientation parameters (roll ϕ, pitch θ, yaw ω angles) tuples.

4.3.1 Baseline Model Parameter Optimization

AVMon predicts the AV states using an Extended (or nonlinear) Kalman Filter (EKF) [59, 57]. EKF is a lightweight algorithm that does not require historical data, using only the previous state information to predict the next possible state of the system. It is capable of solving the non-linear estimation problem through linearizing AV dynamics and output functions for the current estimate to produce an estimate of the next state of the system using Bayesian inference [28].

Its output is an estimate of the joint probability distribution over the variables for each time frame. More information about EKF can be found in Appendix **??**. Our design uses sensory measurements and previously estimated outputs as the inputs to the EKF model to predict the following sensor states. They are compared with the subsequent time frame measurements to detect anomalies. EKF model in our design has two procedures; (1) prediction and (2) correction. The first component takes the last sensors' values estimation and the current sensor readings to generate preliminary predicted sensors' values for the next time step. However, these predicted values have to be refined due to the nonlinear nature of the estimation process. The covariance matrix of the estimation error (i.e., the error between the actual measurements and the predicted states) and the state transition matrix (encapsulating the equations for the vehicle dynamics) are used to obtain the predicted states. Secondly, the correction procedure relies on previous sensors' values predictions, the observation matrix (i.e., a transformation matrix that transforms the AV system from the physical state space to measurement space), and the covariance of the measurements' noises to compute the Kalman gain. Kalman gain is defined as the uncertainty in a predicted state divided by uncertainty in the predicted state plus uncertainty in measurement readings or messages data. Therefore, we get the sensors' predictions that are corrected using the measurement and covariance updated matrices. The outputs of this procedure will be used in the Residual Learning module and will feed the next iteration of this algorithm (i.e., EKF). Next, we need to specify the unknown equations' coefficients to effectively predict the successive states of the AV sensors' values.

To define the dynamics for the AV model and its control algorithm to be used in the EKF for prediction, we have first to use System Identification (SI) [105] to extract the AV control invariants and equations that describe how the vehicle behaves given the control objectives (e.g., a reference position) and the current states. SI derives such equations through regression over a set of collected traces of vehicle operation. Then, since EKF can generate errors in predicting

the dynamic behavior of many systems due to the poor tuning of some of its parameters, such as the covariance matrices Q and R respectively, these parameters have to be tuned to improve the model performance. Thus, we use a Genetic Algorithm (GA) to tune these parameters based on measurement data. Note that this is an offline procedure as shown in Fig. 4.4. GA [16, 119, 85] is a method for solving both constrained and unconstrained optimization problems that are inspired by the Human genetic process of passing genes from one generation to another. We model the AV dynamics prediction competence by evaluating its accuracy based on its coefficients' values. The specific coefficients' values corresponding to the EKF model are coded into a typically binary array, which can be viewed as a chromosome carrying genetic information about the individual, i.e., the EKF model in our case. Thus, GA starts from a widely dispersed initial population of coefficients setups for the EKF model design and converges to the best coefficients' estimation. The pseudocode flowchart of GA is shown in Fig. 4.5.

4.3.2 Residual Learning

Another source of inaccuracy arises because EKF approximates a nonlinear physical system (e.g., the quadcopter) using a piecewise linear process; the prediction suffers large inaccuracies and even filter instability in highly dynamic scenarios [124]. Thus, the prediction can be inaccurate based on two ways: (1) it's physical properties change rate (i.e., position or velocity), (2) how far it drifts away from its real-time behavior due to accumulation of errors. When an AV vehicle changes its velocity slowly, or it moves in a straight line most of the time, the linear approximation will be a good fit, and prediction errors will be lower. However, if the AV vehicle changes its velocity intermittently, the linear approximation will not capture all the velocity variations over most of its domain. Similarly, the further away the vehicle moves from its operating trajectory due to any disturbance, the more likely the linear approximation diverges from the accurate prediction.

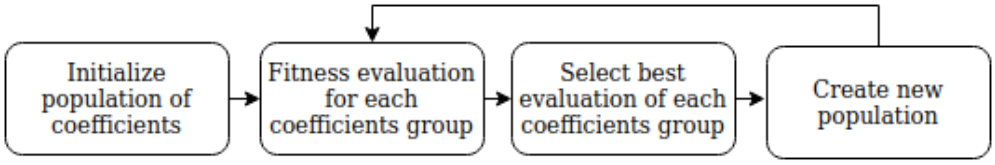


Figure 4.5: The genetic algorithm optimization overview

For more effective detection, we need to minimize the generated error between the predicted and observed measurements. One approach uses a more complex filter that models the nonlinear behavior, but these become computationally prohibitive. Moreover, they may not account for sensors' transient errors and other perturbations that may arise in the real world. Thus, we approach this problem using machine learning to predict the residual dynamics (the expected deviation between the EKF prediction and the measurements). It is important to note that using only data-driven approaches such as machine learning to predict future AV dynamics requires a great source of data to get decent results. When we started our design using only machine learning, we found that its structure is more complicated than using the EKF and a neural network scheme together, and it produces less accuracy and slower response. To learn the residual dynamics, we use a neural network model with ReLU activation to converge faster during training and demonstrate more robust behavior with respect to hyperparameters changes.

As shown in Fig. 4.6, our model includes one input layer, three hidden layers, and one output layer. The inputs for our neural network model include different sensors readings such as the location and orientation of the AV object. At the same time, the outputs are the residual vector per each time frame. The hidden layers include 256, 128, and 64 neurons, respectively. The output layer represents the prediction errors that can be used to validate the outputs coming from the EKF equations. However, designing the critical error-related inputs from a list of all inputs that feed the dynamic model of the vehicle (e.g., roll speed, pitch rate, yaw rate, longitude, latitude, altitude, etc.) is challenging. We solve this problem using a Sequential Forward Selection (SFS) algorithm [51]. SFS is a greedy search algorithm that is used to reduce

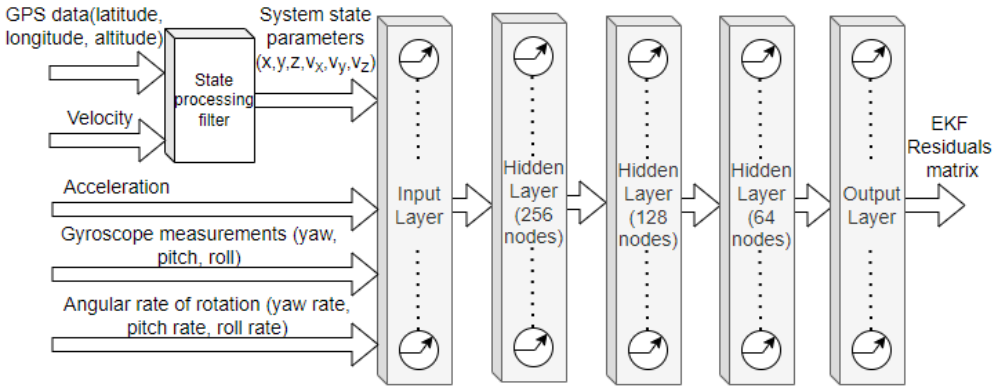


Figure 4.6: The structure of our residual learning neural network module

an initial d-dimensional feature space to a K-dimensional feature subspace, where $K < d$. The motivation behind the feature selection algorithm is to select a subset of the most relevant features automatically. The goal of feature selection is to improve the computational efficiency and reduce the model's generalization error by removing irrelevant features or noises. In our work, we use SFS to add one feature at a time based on the classifier performance for the ANN-based residual prediction model until a subset of features of a size K is reached. Thus, we eventually get a set containing all the essential and most effective K features representing the input layer for learning the residual neural networks model. We start the algorithm by having an empty list of inputs. Then, we add an additional feature, x^+, to our feature subset X_K in each irritation. x^+ is the feature that maximizes our criterion function, that is, the feature that is associated with the best classifier performance if it is added to X_K. We repeat this procedure until the termination criterion is satisfied. The SFS is outlined in Algorithm 2.

4.3.3 Online Anomaly and attack detection

Relying on the instantaneous error value between the prediction states and sensors observation after comparing it to a pre-defined threshold can lead to false positives in the presence of transients and sensor noises. To ensure that only true anomalies are detected, we incorporate a change detection algorithm [117] that is capable of detecting subtle changes in dynamics

Algorithm 2 Sequential forward selection algorithm

A list of the critical features (R) used for our ANN model **Parameters:**
$S \rightarrow$ a whole d-dimensional feature set as input.
$X_K \rightarrow$ a return subset of selected features K, where K<d.
$e_k \rightarrow$ total error when using selected features.
Inputs:
$S = \{s_1, s_2, ..., s_d\}$
$X_K = \{x_j \mid j = 1, 2, ..., k; x_j \in S\}, k = (0, 1, 2.., d)$
$e = \{\infty\}$
Initialization:
$X_0 = 0, k = 0;$

while k < K **do** e^+ = $\text{argmin}_e(e, e(X_k + x^+)) < e, where\ x^+ \in S\text{-}X_k$;
k=k+1;
if ($e^+ < e$) **then** $X_k = X_k + x^+$;
$S = S - x^+$;

providing high confidence decisions. Note that this algorithm applies to the predicted data itself, not as a comparison to the sensor data, to see if the data sequence over time is self-consistent. In our anomaly detection approach, the attacks are detected by first getting the pre-processed sensor readings $\tilde{Y}$(k) to generate the predicted sensor value $\hat{Y}$(k+1) using the EKF algorithm described above. Next, we update the predicted residual e(K) using a neural network so that it is used to compute the final residuals associated with each sensor as follows:

$$r_i(k) = \tilde{Y}(k) - \hat{Y}(k) - e(K). \tag{4.3}$$

Transient errors can be generated from overshoot spikes in the velocity when making a turn due to the simplicity of the Proportional-Integral-Derivative (PID) control algorithm. The PID algorithm determines the control signals based on the error and a weighted sum of the propositional (P), integral (I), and derivative (D) terms. Thus, we should not treat transient errors as an indication of actual attacks. On the other hand, we do not want to miss or delay true attack detection. Our solution is to utilize a function responsible for alerting for the actual malicious activities by computing a statistical detection test [21] that quantifies

the deviation. Specifically, we perform detection by sequentially discounting autoregression time series modeling (SDAR) [96]. In this algorithm, older data values in the sequence are 'discounted', i.e., are less important than more recent values in the sequence. Because recent data is weighted more heavily in an SDAR model, SDAR is well-suited for online change point detection, which focuses on detecting the most recent changes in a sequence. As a result, it outperforms many change detection algorithms and detects malicious values as early as possible.

In AVMon, the anomaly detector algorithm, SDAR, keeps track of the historical changes of the residuals instead of a fixed time window to prevent an attacker from hiding their attack between time windows. In each iteration, new residuals data arrives each time frame with $(t = k+1, k+2, ...)$. Here, we define a parameter S that represents the anomaly score of value. S_k is calculated by updating the mean vector μ and the variances-covariances vector $\hat{\mu}$ as in equation 4.4:

$$\hat{\mu} = (1-r)\hat{\mu}_0 + r * X_t \tag{4.4}$$

Then we calculate S_k through equation 4.5

$$S_k = \hat{\Theta}_i(S_{k-i} - \hat{\mu}) + \hat{\mu} \tag{4.5}$$

where r is the discounting parameter and Θ is a coefficients matrix. Once $\hat{X}_t > \tau_i$, an alarm will be triggered. Finally, the summary of the whole detection block is given in Fig. 4.7.

4.4 Implementation

First, we implement our approach using CARLA simulator [37] that is an open urban driving simulator powered by Unreal Engine (UE) to support development, training, and validation of autonomous urban driving systems and provides open digital assets (e.g. urban layouts,

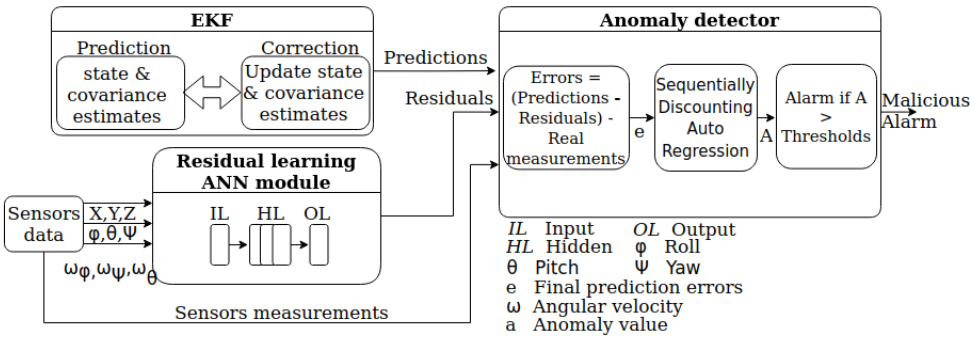


Figure 4.7: The high level structure of the different components inside AVMon

buildings, vehicles). Then, our approach is applied on two different AVs (aerial and ground). Despite both vehicles having different invariants, real-time needs, and specific environments, we show that our methodology can be applicable to AVs in general.

4.4.1 Autonomous vehicles' simulation

In CARLA simulation, the vehicle becomes autonomous by decomposing its driving tasks among perception, planning, and continuous control sub-systems. The perception stack uses semantic segmentation, based on RefineNet classification model [71], to estimate lanes, road limits, and dynamic objects and other hazards. The perception neural network model is trained to classify each pixel in the image into one of the following semantic categories; (road, sidewalk, lane marking, dynamic object, or miscellaneous static). The local planner is based on a state machine with the following states; (road-following, left-turn, right-turn, intersection-forward, and hazard-stop). It is used to synthesize way points that keep the car on the road and prevent collisions . Transitions between these states are performed based on estimates provided by the perception module and on topological in-formation provided by the global planner. Continuous control is performed by a proportional-integral-derivative (PID) controller that receives the current position, speed, and a list of way points to actuate the steering, throttle and brake, respectively. In addition, we create a program that executes our AVMon algorithm based on *Actors* which are

spawned in the simulation by *carla.World*. These *Actors* interact with its sensors via designated APIs (e.g. *Getters* and *Setters* methods) to read sensors' measurements and control vehicle motion.

4.4.2 Aerial AV

We choose Dronecode's open-source PX4 autopilot to simulate our aerial autonomous vehicles experiments. We run the autopilot on a dedicated hardware, Pixhawk 4 [12, 76], which is powered by 32 bit Arm Cortex M7 processor of 216 Mhz, 2MB memory, 512kb RAM. There are also couple of sensors on the Pixhawk board, namely accelerometer, gyroscope, magnetometer, and barometer. We use latest stable version (v1.11.0) of the autopilot. We also made modification on top of this version to run our anomaly detection system, AVMon.

The simulation environment uses Gazebosim [20, 64] for the physics portion of the code. Gazebosim can be used to design a physical world and along with the physics that comes with it. This makes the simulation more realistic. For mission control inputs, we use QGroundControl (QGC) [13]. QGC is a versatile software which can be used in different platform such as computers and mobile devices. Apart from manual control of the drone, it allows users to plan extensive flight missions, such as designating landing location, altitude at which the drone would fly, location of each waypoints, hovering delay at each location, and maximum velocity of the drone between the different points of the flight, etc. All the communication between these components is carried out by MAVLink (Micro Air Vehicle Link). For a run of the simulation, the autopilot code is initiated from the terminal, which also start the Gazebosim. QGC is then started and mission parameters are set. Various data, such as location and orientation of the UAV, actuator controls, actuator outputs, airspeed, CPU and RAM usage, sensor data, etc., get logged by QGC. We use these logs to analyze the flight. We create a module called *reference_monitor* (written in Python) that represent AVMon in the real hardware. On hardware, we use the raspberry

pi 3 model b for *reference_monitor* module with PX4 as a flight controller for QGC. Moreover, in the *reference_monitor*, sensor data, such as GPS coordinates, are per-processed and used for next state prediction so that sensor data can be evaluated as anomaly or not. This module is structured sequentially to introduce a small amount of overhead on the hardware platform.

4.4.3 Ground AV

Our ground vehicle is based on the AVWLtoys A242 model, as shown in fig. 4.8, that is controlled by Raspberry Pi 4 with 2 GB RAM, 1.5 GHz system on a chip (SoC), and a HD camera to stream a reliable frame rate while running the different autonomous algorithms, such as the trajectory planning process. To control the DC motors running towards different directions and different speeds, a motor drive controller board with dual H-bridge DC stepper modules (L298N) is used. Through the Pulse-width modulation (PWM) signals from the (sending) GPIO, the motors drive the vehicle using power from the on board battery via required speed values and directions. The vehicle includes two infrared speed encoders installed on each drive wheel to provide the rotational speed feedback of the wheels, one-dimensional TOF LiDAR (VL53L1X) that is deployed at the front end of the vehicle to detect precisely the longitudinal distance to the closest object in front of the vehicle, and an HD 160-degree fish-eye camera that is deployed on the top of the vehicle. Raspbian is selected as the operation system for the Raspberry Pi board . Raspbian is a Linux distribution based o Debian mostly composed of free and open-source software. The Robot Operating System (ROS) [65] is then applied as the robot platform. ROS is a collection of tools, libraries, and conventions that aim to simplify the task of creating complex and robust robot behavior across a wide variety of robotic platforms. Through the ROS nodes, the distributed control strategy can be utilized, where each node can be a sensor, a computation module, or an actuator. By broadcasting or subscribing to the specific messages, the nodes are able to transmit the information as designed.

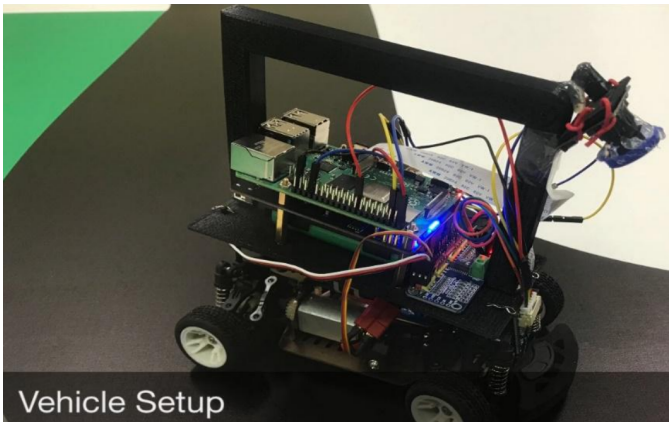


Figure 4.8: Original front wheel steering vehicle

The architecture of the entire system includes four sections: perception, communication, high-level control, and low-level control. The perception is responsible for understanding the environment through the on-board sensors, and then for transmitting the high-level information (e.g., traffic signs ahead) to the high-level control module. Combining the high-level information from both the perception section and the communication section, the high-level control is responsible for transforming the abstract information into detailed instructions such as desired speed and orientation. Finally, the low-level control block calculates the required control signals based on the vehicle dynamics model and sent them to the actuators. Our AVMon algorithm is implemented as a running node within ROS. It receives sensors' measurements from the low-level control node through subscribing and broadcasting pre-defined ROS massages.

4.5 Evaluation

4.5.1 Experimental Setup

In this section, we evaluate our implementation on PX4 autopilot running on Pixhawk 4 and Gazebosim for the aerial vehicle and ROS Kinetic Kame[118] running on Raspberry Pi 4 for the ground vehicle. For the autonomous vehicle simulation, we used CARLA simulator that runs on Windows 10 64-bit with Intel(R) Core(TM) i5-9400 CPU @ 2.90GHz x6 processor

and 16 GB RAM. First, we show the efficiency analysis of the different components in AVMon and how our defense can detect attacks. Then we compare our proposal with other approaches proposed in the literature. Finally, we measure the overhead of our implementation using PX4 for the quadrotor and the AVWLtoys autonomous car.

We implemented our proposed system for the evaluation using python. The experiments were performed on a real data set containing information obtained from different trajectory scenarios. Our experiments are based on maps designed in the autonomous vehicle simulator (i.e., CARLA) for training, testing, and validating. For real experiments, we performed missions to obtain real data-sets containing information obtained from different trajectory scenarios, including simple (i.e., low curvature) and complex(i.e., increased curvature) ones with varying settings of velocity as shown in Fig. 4.9. Different configurations were considered for training the ANN-based residual learning model by changing the number of neurons in the hidden layer, the activation function, and learning rates. For every configuration of ANN, multiple independent experiments were conducted for training, and average results are reported to factor out the stochastic element in ANN network weights' initialization.

To show our model efficiency and generality on a different data set for an AV after training is complete, we tested the resulting ANN-based residual model's weights and biases to show the accuracy of the model on the test data to estimate the accuracy model with new or previously unseen conditions. Once we obtained the highest accuracy, we wanted to ensure that our residual learning model was not over-fitting (i.e., it cannot perform accurately against unseen data, which defeats its purpose). Thus, we used K-fold cross-validation [114], one of the most popular techniques commonly used to detect over-fitting. Moreover, we tested our model against regularized ones using drop out [112] and L2 regularization [82]. Regularization is a technique used to reduce prediction errors by fitting the function appropriately on the given training set. So, we split the data points into five equally sized subsets in K-folds cross-validation,

called ”folds”. One split subset acts as the testing set, and the remaining folds train the model. The model is trained on some trajectories to estimate how the (RL) model generally performs to make predictions on data not used during the model's training. The rest of the trajectories acts as a validation set in each turn. After all the iterations, we average the scores to assess the performance of the overall model. From Table 4.1, we can see that the generated residual learning model is considered the optimal model for generalization among different data sets without any additional over-fitting techniques.

4.5.2 Attack benchmarks

Our attacks were developed as additional software in each system that hijacks the interface between the (actual) control code and the sensor modules. We impersonated the software modules to (1) publish false sensor data (in both PX4 and ROS) to compromise inertial sensors and GPS sensors, (2) control signal spoofing by targeting the control outputs such as the steering from the *PID* controller or the motor pulse width modulation (*PWM*) signals, and (3) do parameter corruption through modifying control parameters (e.g., the *PID* control coefficients) at run time. As an example, the malicious node ($attack\_2$), as shown in listing **??**, replays a chosen image at a higher rate than that of the camera, overwriting any legitimate image with a malicious one and compromising the visual data to affect the steering decisions as shown in Fig. 4.10.

Creating simple attacks by adding bias values stochastically to sensors can introduce significant errors, which is challenging to accomplish without being detected due to the high accuracy of AVMon(although we use some attacks such as these to compare the accuracy of

	Avg.	Standard	W/L2	W/drop out
Quadrotor	Loss	0.00072	0.00624	0.00174
	Accuracy	0.9776	0.934	0.9628
Ground Vehicle	Loss	0.00022	0.00364	0.0037
	Accuracy	0.988	0.972	0.98

Table 4.1: Average K-Fold cross-validation results for the quadrotor and ground vehicle.

((a)) ((b))

Figure 4.9: Sample trajectories used by the quadrotor (a) Simple or low curvature; (b) Complex or high curvature

((a)) ((b))

Figure 4.10: A visual attack on a ground vehicle (a) A real image; (b) An injected image

AVMon to prior work). Consequently, we also consider an advanced attack where the attacker is aware of the internal structure of this detector and attempts to carry out an adversarial attack such that it fools the machine learning component to misclassify the attack as benign eventually. Moreover, we are concerned with a more sophisticated adaptive attacker who attempts to attack while avoiding detection. Thus, we evaluate AVMon against such an attack to show that it is resilient even to powerful adversaries.

4.5.3 AVMon characterization and efficiency analysis

In the first set of experiments, the impact of the individual components of our solution is evaluated as they together track closed loop and straight line trajectories.In a straight line track, using only the default EKF estimator after configuring its relevant parameters can cause an error up to almost 9 meters and an average of 1.77 meters, especially during the turn maneuver for the vehicle. After using GA optimization to configure the EKF, the prediction error becomes significantly lower. However, it experiences spikes of up to 6.5 meters during turns and an average of 1.06 meters. Finally, considering the Residual learning (RL) component further decreases the

prediction error to be less than 0.5 meters at its maximum and 0.16 meters on average (a 10x reduction from just EKF) because it compensates for the non-linear effects that challenge the EKF estimator. For a closed loop or cyclic track, the average error using only the default EKF estimator was not high (0.08 meters). However, the RL component likewise assisted in lowerings the prediction error (0.03 meters). The average and maximum errors of each component are shown in Table 4.2.

Next, we evaluate our anomaly detection performance. In Fig. 4.11, we involved a spoofed wheel speed sensor attack which can be quickly done by injecting a magnetic field. As the attack started at the second ”12”, the signal was maliciously replaced by a constant value. As a result, the ground vehicle lost its control and then tried to compensate for correcting its orientation, potentially causing a crash.

Fig. 4.12 also shows the anomaly score over time using the SDAR detection metric, which clearly and rapidly detected the attack crossing the detection threshold line. The anomaly

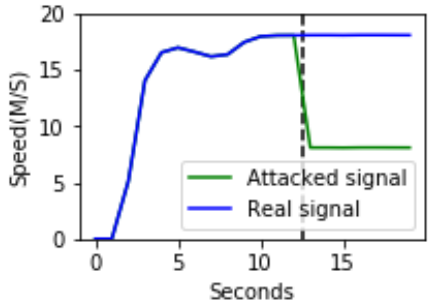


Figure 4.11: An example of a velocity attack on an AV

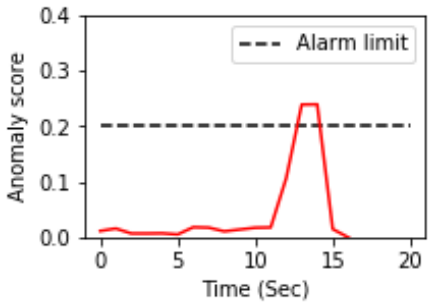


Figure 4.12: Anomaly detection under same attack

	Cyclic track		Straight track	
Components	Avg(M)	Max(M)	Avg(M)	Max(M)
EKF	1.77	8.92	0.08	0.16
EKF+GA	1.06	6.46	0.042	0.08
EKF+GA+RL	0.16	0.49	0.03	0.05

Table 4.2: Average and maximum errors (distances in meters) of main components in AVMon.

score S quantifies the historical deviation based on the SDAR algorithm of the AV system. If $S > threshold$, then an alarm will be raised. Otherwise, the system is normal.[1] Moreover, we launched GPS, IMU, and gyroscope attacks during the quadrotor missions solely to measure the efficiency of our model. As a consequence, AVMon caught all the attacks successfully within an average of less than 0.2 seconds after launching (i.e., zero false negative rate with detection timeliness).

Finally, to depict that our solution does not rely only on the dynamics of the vehicle, we first established solely spoofing attack on GPS position to see the anomaly performance as demonstrated in Fig. 4.13. Then, we added the corresponding injected change to the velocity and acceleration signals to create consistency by fooling the dynamic equations (i.e., vehicle position estimation requires three components: velocity, direction, and acceleration). Nonetheless, the anomaly value got even more significant with the injected values in position, velocity, and acceleration, as shown in Fig. 4.14. We believe that Kalman Gain will decrease if the measurements match the predicted states by analyzing the error covariance matrix. Otherwise, it becomes larger.

4.5.4 Comparison to Savior

In this section, we demonstrate how AVMon improves the security of cyber physical systems by comparing it to Quinonez et al.'s recent defense, Savior [95], under a range of scenarios and attacks. SAVIOR algorithm predicts physical observations based on EKF for

[1] A video demonstrating the attack can be found at this link.

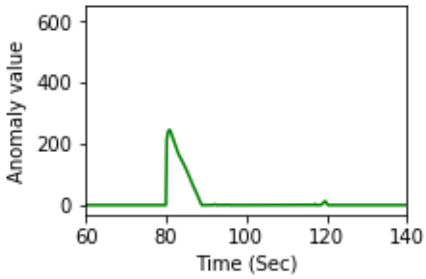


Figure 4.13: Anomaly value over time after the spoofed attack in latitude

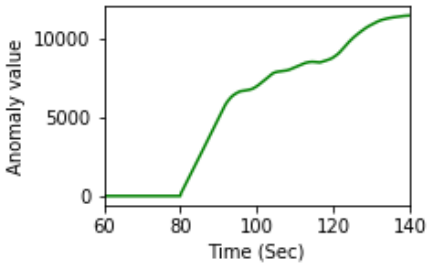


Figure 4.14: Anomaly value attacking latitude, velocity, and acceleration simultaneously

predicting the observations and cumulative sum (CUSUM) [80] algorithm for anomaly detection We performed a series of experiments comparing AVMon and SAVIOR using CARLA simulator, ground, and aerial vehicles.

Prediction performance

First, both predictions can predict well in most cases due to using nonlinear prediction filters, which are suitable for the nonlinear dynamics of the AV. However, SAVIOR has larger prediction errors. On the other hand, AVMon can accurately predict the system states even when there are quick changes in the trajectory of an AV as depicted for in Fig. 4.15 and Fig. 4.16.

Then, to study both schemes under different uncontrolled conditions to realize under what circumstances AVMon can be better and under which ones will be worse or more expensive. We tested both schemes on the quadrotor under different parameters, including various trajectories' curvatures, velocity, multiple stopping in midair, etc. We used the quadrotor in this test since it has more movement ranges or dimensions. From Table. 4.3, we can see that AVMon improves

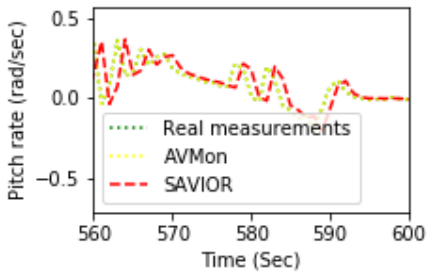


Figure 4.15: Pitch angle prediction comparison

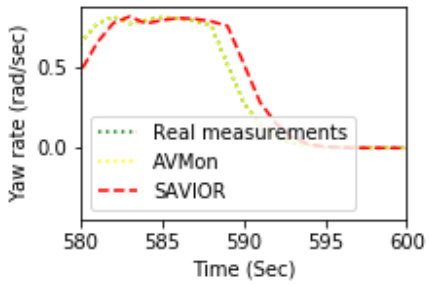


Figure 4.16: Yaw angle prediction comparison

the prediction quality slightly over previous schemes on straightforward trajectories. However, it significantly improves the prediction quality in high dynamical or curative routes.

Next, we launched different bias attacks that are injected into the location reading of the ground vehicle, and we measured if the anomaly detector in both schemes is capable of detecting the attacks for different intensities, as shown in Table. 4.4. The results show again that AVMon outperforms SAVIOR with better precision, so faster detecting attacks can be accomplished.

The previous examples show that our system can detect attacks efficiently. Still, the question is now, what if both schemes did not detect an attack and considered it in their calculations (i.e., worst-case scenario)? How long will the system be unstable? The stability of a state estimator [49] is typically defined in terms of the convergence of the state estimate to the normal state, or, in other words, the state estimate error converges to zero or becomes bounded within some region near zero. Thus, we analyzed all the sensors' attacks on AVMon as shown in Fig. 4.17, and we found out that roll angle has the most impact on the system stability. We

Curvature	Velocity	Mid-air stops	AVMon accu-racy	SAVIOR accu-racy
High	High	no	98.77%	89.19%
High	low	no	100.0%	95.03%
High	low	Yes	99.44%	88.72%
low	High	no	100.0%	97.62%
low	low	no	100.0%	98.26%
low	low	Yes	100.0%	97.58%

Table 4.3: Prediction accuracy comparison over different routes.

Attack bias range (M)	AVMon	SAVIOR
< 1	detected	Not detected
3	detected	Not detected
> 5	detected	detected

Table 4.4: Detectability comparison using different bias values in the location readings for a ground vehicle.

compare the stability under different bias attacks in the roll angle generated by the gyroscope of the quadrotor by computing how long the systems stay faulty or unstable as shown in Fig. 4.18 and Fig. 4.19. From the results, it is noted that AVMon performs slightly better than SAVIOR for low curvature trajectories. However, errors drop dramatically in AVMon and the system stability can improve up to more than %70 in some cases in high curvature trajectories.

Time-To-Detect (TTD is an essential metric for cyber physical systems because detection delays can provide an attacker an opportunity to cause significant damage. Thus, we

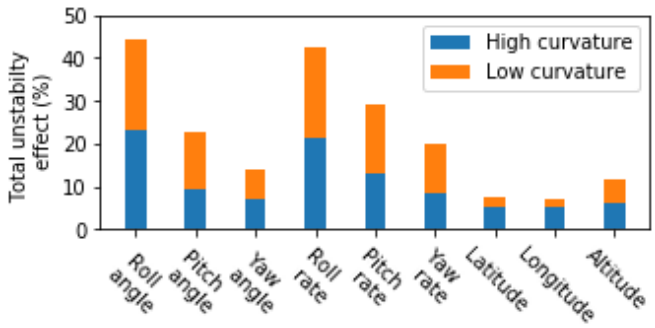


Figure 4.17: Sensors' attacks effect on the state estimator stability

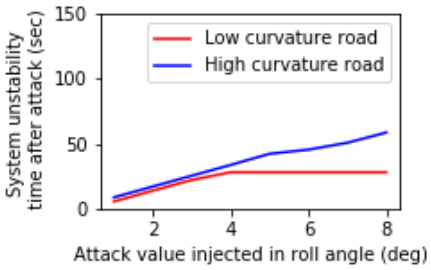


Figure 4.18: Stability effect by roll angle bias attack under AVMon

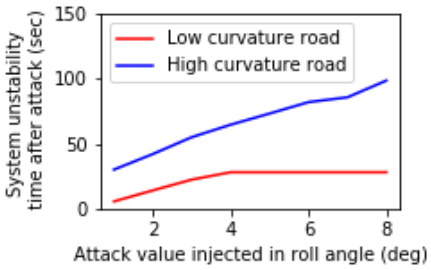


Figure 4.19: Stability effect by roll angle bias attack under SAVIOR

launched various bias attacks injected into the AV sensor signals to compare TTD performance. Fig. 4.20 shows a bias attack injected into the quadrotor gyroscope, especially into the pitch angle rate (i.e., angular velocity over the Y-axis), and we measured the time it takes to detect the attack for different intensities. As a result, AVMon can catch this attack faster than SAVIOR by (0.2 seconds). SAVIOR takes longer to detect such an attack due to the significant prediction errors from its algorithm, which requires larger margins to reach the desired anomaly detection threshold without causing excessive false positives. On the other hand, AVMon uses a nonlinear states estimation filter with the help of the residual learning model to improve prediction accuracy so that the change detection component (SDAR) detects the attack quicker.

An accurate position estimation is perhaps the most critical component of establishing context-awareness in AVs. Therefore, we show the position error (i.e., the difference between the measured position of the AV system and the next time frame predicted position based on x, y, and z coordinates) by carrying out two experiments: (1) a ground vehicle with a simulated trip in CARLA simulator; (2) a quadrotor using a real data set [29] of a Parrot (Bebop 2) quadrotor as it

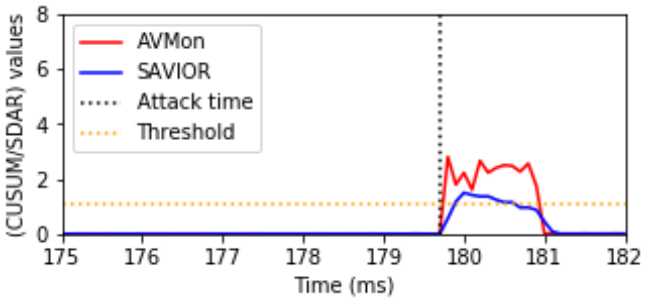


Figure 4.20: Time to detect (TTD) for AVMon and SAVIOR against the same attack

captures videos; the data set includes comprehensive sensor readings including current timestamp, GPS, and IMU sensor data. Fig. 4.21 shows that AVMon outperforms SAVIOR in positional prediction for the ground vehicle with errors less than 0.3 m, whereas SAVIOR reaches over 1 meter error in some cases. Fig. 4.22 shows the predicted position errors for the quadrotor, staying less than 0.5 m for AVMon but fluctuating up to 4 m using SAVIOR. The higher accuracy gives the attackers less opportunity to hide within the prediction envelope. Moreover, the stability of the prediction allows the detection thresholds to be tightly set without generating false positives.

Furthermore, we experimented with different spoofed values for derivative coefficient K_d and integral coefficient K_i for the *PID* controller to cause the AV to get out of its reference trajectory. K_d reduces the overshoot caused by the proportional component. K_i fixes the system-

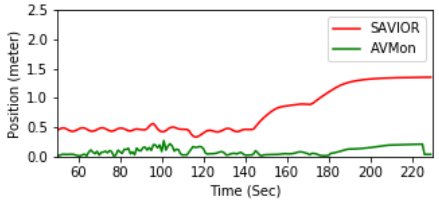


Figure 4.21: Positional error comparison in a ground AV

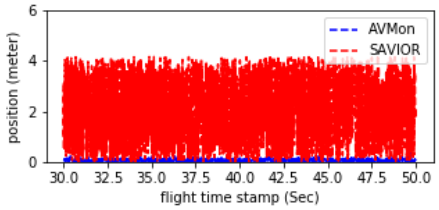


Figure 4.22: Positional error comparison in a Parrot (Bebop 2) quadrotor

atic bias caused by the steering angle over time, which could eventually drive the vehicle out of the track. We also spoofed the steering angle output from the *PID* controller. As a result, we can see that AVMon has faster TTD action than SAVIOR as depicted in Fig. 4.23 and Fig. 4.24.

End to End Attack Detection

We evaluate the end-to-end performance of the AVMon compared to Savior. For each detector, a threshold needs to be chosen on the anomaly score to balance sensitivity to attack against the probability of false positives. When we decrease the threshold, we get more positive values. Thus, it increases the sensitivity but decreases the specificity. Similarly, if we increase the threshold, we get higher specificity and lower sensitivity and potentially increased TTD. To tune the detection thresholds, we empirically used a large number of diverse experiments, including those with complex trajectories, to plot the Receiver Oriented Characteristics (ROC) curve, which characterizes the performance of the detectors for different detection thresholds. Thus, we picked suitable thresholds that balance between sensitivity and specificity. From Fig. 4.25, we observe that: (1) with the same threshold, AVMon outperforms and has lower FP rates, (2) by selecting

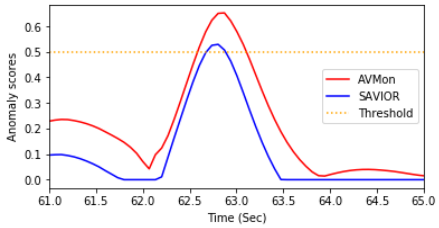


Figure 4.23: Anomaly performance with high injected K_d value

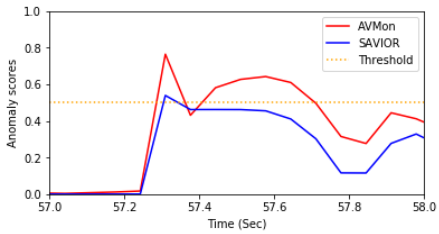


Figure 4.24: Anomaly performance after spoofing steering

smaller threshold values, we can detect attacks faster due to the smaller FP values, (3) AVMon can detect the attack with a probability close to 100% while having an FP rate (below 1%) using the same thresholds values for detecting the anomaly values as SAVIOR uses. To achieve the same sensitivity, SAVIOR would have over 30% FP rate.

Resiliency to stealthy adversarial attacks

Here we developed a stealthy adversarial attack that injects adversarial perturbations to avoid detection. Specifically, we sampled the system operation over N training samples and used these samples to obtain a loss function to produce adversarial turmoils with small magnitudes representing an attack without being detected. In this attack, we predict the target sensor reading $Y_i(k)$, such as altitude, to calculate a loss function SQR_err as follows:

$$SQR\_err = (Y_i(k) - Y_i\hat{(k)})^2 \tag{4.6}$$

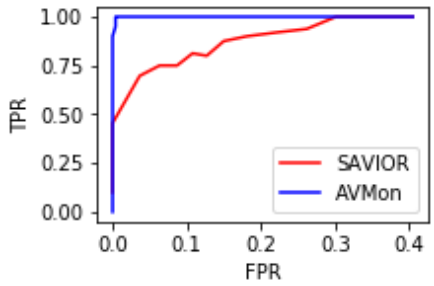


Figure 4.25: ROC comparison implemented in a quadrotor

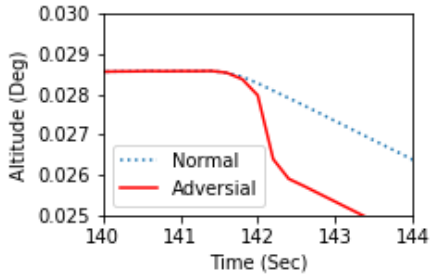


Figure 4.26: Adversarial attack targeting altitude

The next step is to compute the gradients with respect to *SQR_err* as follows:

$$gradient = -2 * \Sigma(sq\_error)/N \tag{4.7}$$

where N is the number of sampling iterations. Then, the adversarial perturbation that will be added to the sensor value will be calculated as follows:

$$Y_i^{'}(k) = Y_i(k) + \varepsilon * gradient \tag{4.8}$$

Fig. 4.26 shows an example of an adversarial sensor attack that is injected in the GPS altitude measurements with an injected attack using the Fast Gradient Sign Method (FGSM) attack [90]. In this example, ε is set to 0.025 to ensure the perturbations are small. Using AVMon this attack was detected quickly, while it is hard for SAVIOR to detect it. In this attack, the adversarial attacker was employed to simulate a crash for a quadrotor by targeting the yaw angular velocity and the GPS altitude position while landing. The quadrotor failed to land smoothly after passing the altitude position (0.0285) since the attack produced smooth deviations in the Z direction, and the controller was trying to compensate for the misleading information. We tried various epsilon values to make this study systematic since it is a critical parameter for this kind of attack. Lowering epsilon can generate a slower attack that eventually may not be detected. Note that from Table. 4.5, the attack was detected efficiently with AVMon and SAVIOR at a large epsilon value. Nonetheless, AVMon manages to track the adversarial attack under smaller epsilon values while SAVIOR could not. This experiment shows the importance of considering such attackers on physical-based control systems for autonomous vehicles. Eventually, AVMon stops detecting this attack with ($\varepsilon = 0.0025$) since the adversarial attack has a lower anomaly score than the defined threshold.

Epsilon value for the attach	Attack detected by AVMon	Attack detected by SAVIOR
0.50	Yes	Yes
0.25	Yes	No
0.025	Yes	No
0.0025	No	No

Table 4.5: Anomaly detection for different epsilon values.

Sensitivity to external disturbances and noise

In real-world deployments, external disturbances such as weather, road conditions, or system degradation can occur outside the dynamics model used in the detector. To study the effect of these disturbances, we measured the prediction errors under different wind speeds and environmental scales (e.g., clear, overcast, and rain conditions). Even though these effects are not included as parameters in the physical model, they can interfere with the ability of a model-based predictor such as EKF to predict accurately. As the prediction quality degrades, the system can produce higher false positive errors. Specifically, significant weather conditions, such as wind, can affect an AV by pushing by exerting forces that change their dynamics (position or angular position) or introducing noise to specific sensor modalities. However, the PID controllers compensate for these effects, especially if the weather conditions are not extreme (i.e., wind speed is around less than 5 m/s). Under such conditions, the prediction quality of the EKF predictor degrades with effects outside of its model. However, the residual learning algorithm in AVMon can compensate for these external disturbances and therefore experience substantially lower false alarm rates than SAVIOR, as shown in Fig. 4.27.

To more systematically study this effect, Fig. 4.28 shows the FP rates results for AVMon and SAVIOR using injected Gaussian noises with increasing standard deviation values as shown on the x-axis. AVMon outperforms SAVIOR substantially.

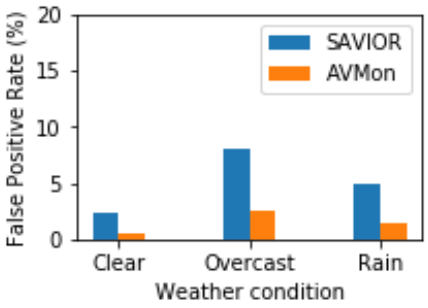


Figure 4.27: False positive rates under different weather condition

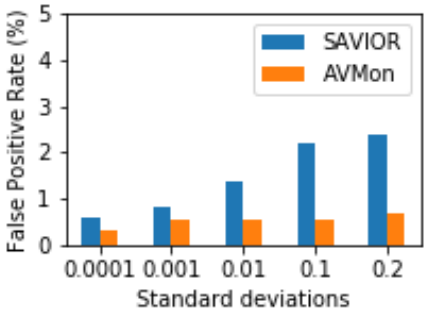


Figure 4.28: False positive rates for SAVIOR and AVMon in aerial vehicle for different noise levels

4.5.5 Performance Overhead

Now, we explore the performance overheads of AVMon. In particular, for practical deployment, the different components of the system must have low overhead to enable decisions to be taken within the real-time feedback loop of the AV controller. We implemented AVMon in the autonomous ground vehicle using a single collection of *TensorFlow*[14] checkpoint files to represent the residual learning model requiring 225.5 kB in total storage. We added our module to the Robotic Operating System (ROS) controller running in the vehicle as a ROS node executed as a session. We measured the execution time of the AVMon module (e.g., conducting its *Path planning* algorithm) for 5 minutes using two different tracks. Time measurement commands were injected between the start and end of each control loop iteration to record the execution time for each iteration. Then, the average execution time metric was calculated as an indication for evaluating the runtime performance. The overhead represents the average execution time with

respect to the real-time constraints or timestamp (i.e., each 0.1 sec). So for the ground vehicle, Table 4.6 shows that the average execution time of our module is less than 0.01 seconds with %4.1 overhead, on average. As for CPU utilization and memory (i.e., capacity overhead), AVMon consumes on average %7.0 CPU usage, as shown in Table 4.7. Of course, this number can be even smaller with higher hardware specifications than the Raspberry Pi we used in this experiment or reduced ROS cycles per second value (currently, we operate at 10 times per second). Note that using a Raspberry pi based ground vehicle is common in academia. However, working with smaller size autonomous vehicles is challenging since trajectory planning control algorithms perform with more stability in more oversized vehicles. For our implementation in the aerial

	Quadrotor	Ground vehicle
Without residual learning	0.0075 sec	0.0013 sec
With residual learning	0.012 sec	0.0041 sec
Computational overhead	%10	%4.1

Table 4.6: Average execution time and overhead for the quadrotor and ground vehicle.

AV type	%CPU	%MEM
ground vehicle	%7.5	%8.0
quadrotor	%7.5	%3.0

Table 4.7: CPU and memory utilization for the AVMon module.

vehicle, we used the latest stable version of a popular drone operating system, PX4 v1.11.0.The size of our module code is 8.5 KB. AVMon module utilizes a library called Dronekit to get general information from it, including telemetry data, sending action commands like arming, take off, land, etc. It starts by calling the $connect()$ method. Then, get/set parameters and attributes are used to control the quadrotor movement. The execution time of the AVMon module, for the aerial vehicle, was, on average, 0.012 seconds with only %10 overhead, as shown in Table 4.6. In addition, the module consumes %7.5 of the CPU time (Table 4.7). Besides, we did not observe any latency during our module execution over 0.1 seconds during our experiments.

Chapter 5

Mitigating Application Attacks on Connected Vehicles

5.1 Threat Model

We consider a CV environment where attackers can compromise a device that is eligible to obtain a certificate from SCMS and participate in the system. SCMS prevents an attacker from spoofing other CVs' identifiers. However, our concern is with those attacks where malicious actors participate in CV protocol by replaying valid messages or sending them with fabricated data. In this study, we do not consider attacks that target message delays, jamming, physical attacks on sensors or controllers. We also do not consider mitigation against attacks that exploit bugs in the software stack of any existing components running on the infrastructure or vehicles.

We assume that vehicles communicate through Dedicated Short Range Communication (DSRC) devices based on IEEE 802.11p[61]. However, the proposed system does not limited to DSRC. Equipped vehicles with onboard units (OBUs) can send Basic Safety Messages (BSMs) to other equipped vehicles or infrastructure consisting of two parts: BSMs-Part 1 are transmitted

periodically (typically every 100 msec), containing critical data elements such as vehicle size, position, speed, heading, acceleration, brake system status. BSMs-Part 2 are event-based messages, containing various optional data elements customized by different manufacturers, e.g., ABS activated.

On the other hand, Road Side Units (RSUs)are used to coordinate behaviors across cars or to maintain certain shared states.

We consider the following types of attacks.

- **Fake Message Attack** where the adversary starts to create messages' fields or parameters such as velocity, position, and acceleration, with injected biased values, and then broadcasts the messages to harm the entire system.
- **Replay Message Attack** where the adversary receives and stores a beacon that is broadcast by the other vehicle, and replays it at a later time with malicious intent. The replayed beacon contains old information that can lead to hazardous effects.
- **Stealthy Message Attack** where the adversary is aware of the vehicle dynamics and hence employs its parameters to avoid normal detection filters that can easily find random malicious behaviors. To create the position field in this attack, we use the following equation:

$$lp_{t+1} = lp_t + s_t * \Delta_t + a_t * \Delta_t^2/2$$

where lp, s, a, t, and $t+1$ are the lane position, speed, acceleration, recent time step, and next time step, respectively. This equation calculates position using speed, acceleration, and the time difference between the recent and previous times. The vehicle's speed increases with a_t at each time step until it reaches v_{max}. The vehicle's speed cannot exceed the speed limit of the roadway segment.

5.2 CV Applications

In this section, we highlight some of the CV applications that are used as case studies in this paper.

In **Intelligent Traffic Signals** (I-SIG), RSUs host I-SIG to manage intersections adaptively and intelligently with goals such as reducing collision, decreasing idle time, and improving traffic flows. The BSM messages are broadcast by CVs and captured by a trajectory awareness process that sustains the latest trajectory for each vehicle linked to its vehicle ID. The signal planning process monitors the traffic signal status. Then, it develops a signal plan based on the incoming real-time trajectory data that are fed into the COP (Controlled Optimization of Phases) algorithm[100][39]. After planning, the I-SIG controller sends signal control commands to the controller. The COP algorithm estimates each vehicle's arrival time and uses dynamic programming to calculate the optimal signal plan with the least (estimated) total delay.

Cooperative Ramp Merging System: In Cooperative Ramp Merging, the V2I system localizes vehicles on a virtual map to be used later by different control algorithms to provide driver assistance to enable safe and smooth merging. The output of the merging control algorithm contains the recommended speed for any cooperative vehicles in the merging process. In particular, the reference accelerations are calculated by a feedforward/feedback control algorithm [128] and sent to the vehicles, so they can regulate the gaps smoothly even before reaching the merging point. As a result, rear-end or sideswipe collisions in the conflict zone are essentially eliminated. Finally, **Cooperative Adaptive Cruise Control** (CACC) utilizes V2V communication to form platoons of vehicles that travel with closer spacing, reducing aerodynamic drag, and improving roadway capacity[113, 93, 77]. In this implementation, the Platoon Management Protocol (PMP) [23] controls platoon operations and maneuvers. The leading vehicle acts as the coordinator and controls platoon maneuvers such as speed, lane change, and merging with other platoons.

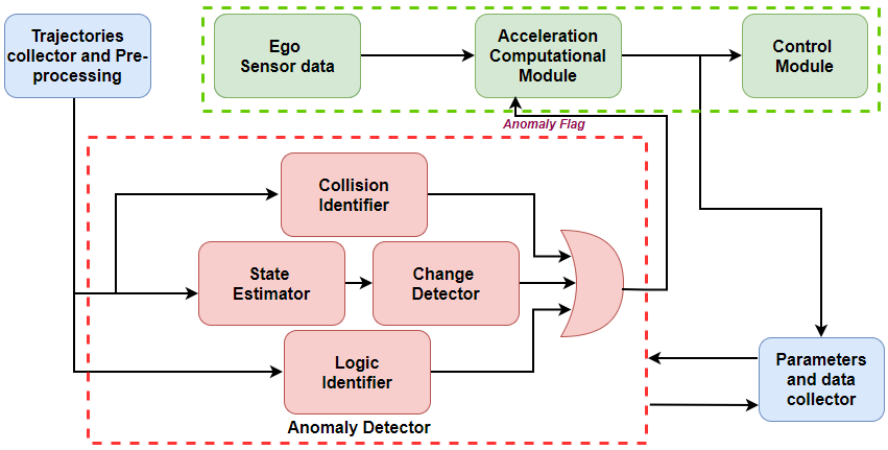


Figure 5.1: An example of CVGuard architecture used in CACC

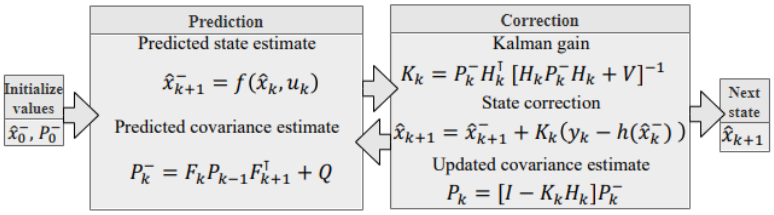


Figure 5.2: The state estimation algorithm

5.3 CVGuard architecture

Application-level attacks rely on communicating falsified information to manipulate a car or a group of cars into an attacker's desired action. The essential question to defend against such attacks is how to determine whether or not the information received from remote CVs is trustworthy. We break down this problem into three smaller problems: (1) Is the predicted state of the car consistent with the measured state? (2) Does acting on the received information potentially lead to collisions or other dangerous actions? And (3) Are the received application-level actions consistent with the application's logic (e.g., do they reflect valid maneuvers?). The components of our defense try to answer these questions, respectively, to detect application-level attacks from different perspectives: when they deviate from the application logic; when they cause the measured and predicted states to diverge; and when they lead to dangerous situations such as collisions. Figure 5.1 gives an onboard architecture of our proposed CVGuard system. It

consists of: (1) **State Estimator**; (2) **Change Detector**; (3) **Collision Identifier**; and (4) **Logic Identifier**. In the *State Estimator* algorithm, external attacks can be potentially detected by checking whether the (perceived) physical states of the vehicle are consistent with its expected states determined by its dynamics and control. It is defined by the dynamic system properties and control algorithm, mathematically represented by control invariants. The Change Detector can detect unusual changes in states based on a cumulative sum of recursive residual statistics. It allows the system to detect attacks accurately and rapidly, avoiding false positives due to transient errors. Even though individual messages seem to be acceptable by the *State Estimator* algorithm, they can still lead to collisions. This could be caused by stealthy attackers who launch attacks that maximize the damage to the system without being detected to cause a crash or cause other safety problems. The *Collision Identifier* predicts the location of nearby vehicles to detect potential collisions or other hazards. Finally, the Logic Identifier ensures that a protocol or maneuver output does not compromise the safety even under attacks, which relates to under-specified or incomplete protocol logic. For example, if an application logic fails to consider corner cases, such as a large gap between two platoons before merging, CVGuard considers it a V2V anomaly. The logic identifier is specific to each application.

5.3.1 State Estimator

An accurate state estimator that tracks vehicles with changing dynamics can be achieved by using multiple filter models that provide estimates of some variables such as velocity and acceleration. Our state estimator adopts an *Interacting Multiple Model* (IMM) [75] that estimates updated states and state covariances based on the ensemble of the most common models (e.g., constant velocity, constant acceleration, constant turn, etc.). Every model is a single Kalman filter that re-initializes with mixed state estimates and covariance based on their probabilities of ”switching to” or ”mixing with” each other. Thus, constantly correcting each filter to reduce its

residual error, even when it does not represent the true motion of the object. In this way, an IMM filter can switch to an individual model based on the specific vehicle dynamics without waiting for convergence first. Thus, using these models can adequately predict the tracked vehicle's possible motions, which is better than using only one model over time.

The different models of the IMM, such as constant velocity or constant acceleration models, follow the same steps of the extended Kalman filter. However, they differentiate in using the dynamical equations in the "predict state". In general,the algorithm [108] is divided into two main procedures: prediction and correction. The first component takes the last estimation $\hat{x}_k$ and the current input u_k, and generates a prediction $\hat{x}_{k+1}$. However, this prediction is refined using the received data. Similarly, the covariance matrix of the estimation error P_k^- (i.e., the error between the real states x_k and the estimated states $\hat{x}_k$) is predicted using the process covariance matrix Q and the state transition matrix F_k. The second procedure uses the previous predictions $\hat{x}_k^-$, P_k^-, the observation matrix H_k, and the covariance of the measurement noise V, to compute the Kalman gain K_k, which is defined as the uncertainty in a predicted state divided by the uncertainty in predicted state plus uncertainty in measurement readings or messages. Therefore, the state prediction is corrected using the measurement and the covariance matrix is updated. The output of this procedure $\hat{x}_{k+1}$ and P_k will feed the next iteration of the algorithm as $\hat{x}_k^-$ and P_k^-.

The inputs of the EKF can be the position and velocity in a time step. At the same time, the outputs can be the estimated position and velocity that the vehicle may happen in the next time step. The goal of using the EKF is to combine any instantaneous reading (which could be maliciously injected values) with the observed dynamics of the system, allowing us to identify unreasonable/inconsistent measurements rapidly. The EKF structure is shown in Fig. 6.1.

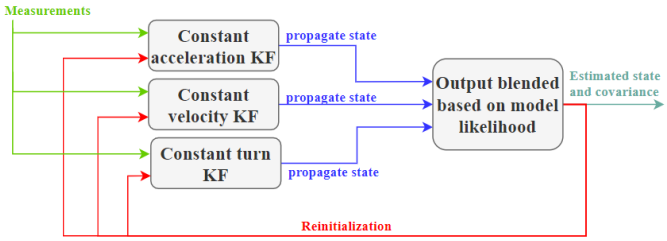


Figure 5.3: Overview of the State Estimator phase

5.3.2 Change Detector

The change detector is essential to detect errors from compromised (or noisy) sensors by characterizing the signals changes and controlling the overall error rate. Thus, we propose using the Cumulative Sum of Recursive Residual (CUSUM) [81] change detector that continuously monitors the error of the regression model. A significant increase in the error is interpreted as a change in the distribution that generates the examples over time. When a change is detected, the actual regression model is deleted, and a new one is constructed. The CUSUM statistic is described by the following recurrent equation:

$$S_i(k+1) = S_i(k) + |r_i(k)| - b_i \tag{5.1}$$

Where r_i is the prediction residual associated with each sensor and S_i is the anomaly score. $S_i(0) = 0$ and $b_i > 0$ are selected to prevent $S_i(k)$ from increasing when there are no attacks. When $S_i(t_k) > \tau_i$, an alarm associated with sensor i is triggered where τ_i is the threshold value.

5.3.3 Collision Identifier

The state estimator is a lightweight solution that efficiently detects cyber-physical attacks on an individual CV. However, we need to identify and avoid potential collisions and other undesirable conditions. Thus, we use Reinforcement Learning (RL) [83] to learn the proper maneuver sequence that can help detect anomalies even if the dynamics of each connected vehicle

appear to be accurate. The *Collision Identifier* RL algorithm is based on three components: state, action, and reward. The state describes the current condition of an agent. The action is what the agent can do in each time step. Finally, the reward describes positive or negative feedback from the environment due to the agent's action. The overall goal of RL is to learn a policy that maximizes the total reward of an agent through learning from the states and actions when it interacts with the environment. In our *Collision Identifier* algorithm, we use Q-learning [115, 103] because of its combination of effectiveness and simplicity. Q-learning is a value-based learning algorithm that updates its value function based on the Bellman principle. First, we create a Q-table where the agent can update its item by learning the rewards associated with all state-action pairs, based on the following equation at each time step:

$$Q^{new}(s_t, a_t) = (1-\alpha).Q(s_t, a_t) + \alpha \times (r_t - \gamma.max_a Q(s_{t+1}, a))$$

where α is the learning rate (ranging from 0 to 1) and represents how much the agent should learn from a new observation; r_t is the acquired reward for any taken action; γ is the discount factor that controls how much each reward can affect our decision; $max_a Q(s_{t+1}, a))$ is the estimated reward from the next action where the agent selects the optimal action to maximize the reward; $Q(s_t, a_t)$-values are the estimated values, and they represent how much the agent expects to get after performing an action. s is the current state of the vehicle. Since the goal of RL is to maximize the long-term rewards through maneuvers, we design the state, action, and reward as follows:

State: We design our states as a set of possible situations where the vehicle can inhabit during movement. These states represent also neighbor vehicles around the vehicle of interest (also called ego vehicle) during an action, and its nearby detected vehicles are called remote vehicles. In our model, we consider a total possible number of the ego vehicle's states or s_t of 16 discretizing

the different configurations of nearby vehicles. For example, s_1 represents an ego vehicle that has only one vehicle in front of it.

Action: We define the action space $A = \{action_1, action_2\}$ to refer to the two possible action patterns of an agent in the system, reflecting either normal or abnormal behavior. The action classification is based on **Time-to-Collision (TTC)** while observing the CV traffic system; if TTC deviates outside preset threshold reflecting normal operation, then we consider the system to be anomalous (i.e., $action_2$); otherwise, we consider it normal ($action_1$). The action is used in the Q-table to define the probability that the agent takes a normal behavior or not. **TTC** is a metric [78] that measures the time taken for a vehicle to collide with the vehicle in front of it, which is an important metric to measure how safe CV components are under cyber attacks quantitatively. TTC of vehicle i at instant t can be calculated as follows,

$$TTC_i(t) = \frac{D_i(t) - D_{i-1}(t) - l_i}{V_i(t) - V_{i-1}(t)}$$

where $D_i(t)$ and $V_i(t)$ stand for the position and speed of vehicle i, respectively, at instant t, and l_i is the length of the vehicle i.

Reward: We design our reward scheme to have a minimal positive value for safe actions and a large negative value if any safety violation occurs. The reward value is determined by TTC of the agent or ego vehicle, as shown below. The $Threshold$ value is estimated throughout experiments to be 0.5 seconds.

$$r_t = \begin{cases} 1, & \text{if } TTC > Threshold. \\ -10, & \text{otherwise.} \end{cases} \tag{5.2}$$

The RL algorithm is shown in Algorithm 3.

Algorithm 3 Reinforcement learning training process

```
1: procedure UPDATE Q AND CALCULATE LOSS FUNCTION
2:     Initialize Q and S values
3:     while S_t is not terminated do
4:         if A_t violates the reward policy then
5:             R_t = big negative reward
6:         else
7:             R_t = small positive reward
8:         end if
9:     end while
10: end procedure
```

5.3.4 Logic Identifier

To ensure that a CV protocol or maneuver does not compromise the safety of the included CVs under attack, we have to practice a plausibility check functionality or safety policies. To achieve this, it is necessary to start with a systematic study of the main properties of each CV maneuver since the discovery of such characteristics can most generally affect the security of their corresponding implementation instances. In Algorithm 4, we develop a simplified protocol in which the model updates the timer according to the event. This event represents all the known CV events or maneuvers. Every event in $EventRange_i$ triggers the same update on $timer[i]$. The $Retrieve$ function reacts with the CV environment to pick different properties if such a property is available. These properties assure the CV protocol's safety and include space gaps, relative locations, relative velocities, lane consistency, etc. For example, in the reference CACC implementation, an anomaly can be generated if the space gap does not reach the value below the maximum safe threshold. Thus, ensuring the robustness and safety of the protocol algorithm under the different application-level attacks.

Algorithm 4 A simplified Logic Identifier Algorithm

```
1: EventRange = (0 ··· (N − 1))
2: TimerIndexRange = (0 ··· (M − 1))
3: Property = (0 ··· (P − 1))
4: while number of CVs ≠ 0 do
5:     for ∀ event ∈ EventRange do
6:         timer = [i ∈ TimerIndexRange ↦ None]
7:         if timer[i] > 0 then
8:             timer[i] − 1                                  ▷ count down
9:             RETRIEVE(Property)
10:        else
11:            TIMEOUT                                       ▷ expire
12:        end if
13:    end for
14: end while
```

5.4 Evaluation

5.4.1 Experimental Setup

To evaluate CVGuard, we used VENTOS (VEhicular NeTwork OpenSimulator) [7], which is an extension of Veins simulation [110]. VENTOS enables us to design, test, and evaluate different traffic scenarios. It integrates a C++ simulator for studying vehicular traffic flows and combines two widely used simulators, Simulation of Urban Mobility (SUMO)[4] and OMNET++[2]. SUMO is an open-source road traffic simulator that uses Traffic Control Interface (TraCI) to communicate simulation commands.

5.4.2 CVGuard Effectiveness

One of the most challenging aspects of using simulator data is fidelity. Unfortunately, no repository of sufficient real-world driving data from various driving scenarios is available. To address this issue, we monitored the vehicles data over time coming from the road traffic simulator (SUMO) and compared it with the HighD dataset [66] that provides trajectory data corresponding to actual vehicles driving on German highways. We validated that the SUMO data distribution is consistent with HighD with several metrics. We carried out the following

experiments to demonstrate how we built and configured the state estimation component of CVGuard. Different vehicular dynamics characteristics can be used to create the IMM algorithm. However, using a large number of models impacts performance negatively. Thus, we wanted to build an IMM algorithm with the best-suited and most efficient configurations under different trajectory segments while driving over time to improve the tracking accuracy and model switching speed of maneuvering target tracking. In the simulation, we tested different vehicles and measured metrics such as velocities and steering behaviors, as shown in Figures 5.4, 5.5, and 5.6. These three selected dynamics characteristics show that the vehicle switches mostly its dynamics between constant velocity, constant acceleration, and turning right or left to reach its destination. As a result, three Kalman filter models have been selected to create and test the IMM algorithm based on the analysis of the dynamics characteristics evaluation. These models are a constant velocity (CV), a constant acceleration (CA), and a three-dimensional turn with a kinematic constraint (TURN) Kalman filters. Adding additional models to IMM can lead to overfitting problems and degrade detection performance.

To evaluate this phase, we made the malicious vehicle manipulate its parameters and broadcast its forged or fake messages to the nearest connected vehicles. These BSM messages can contain false synthetic variables such as velocity and acceleration. Thus, the state estimator filter produces noticeably significant prediction errors. Therefore, it can accurately predict the

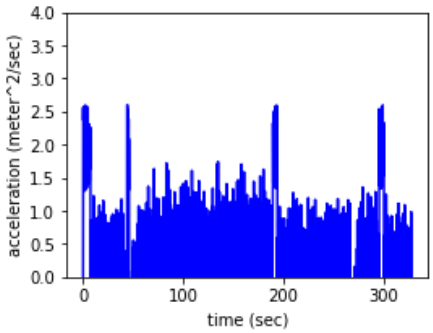


Figure 5.4: A constant acceleration (CA) example

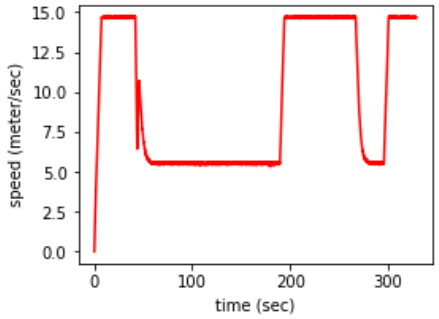


Figure 5.5: A constant velocity (CV) example

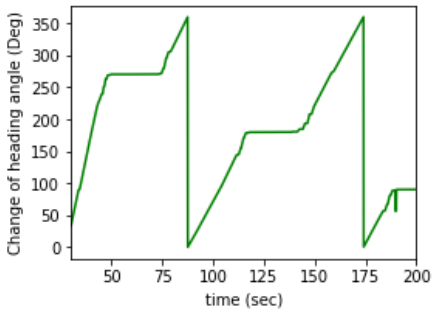


Figure 5.6: Turning with a kinematic constraint (TURN) example

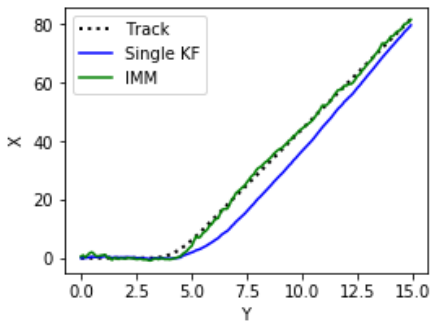


Figure 5.7: Improved tracking performance using IMM filter

surrounding connected vehicles' states, as shown in Fig. 5.9 and Fig. 5.10. To detect the presence of the cyber-attacks and filter out the transient errors caused by physical disturbances (e.g., winds), we use a change detector component that is based on the non-parametric cumulative sum (CUSUM) anomaly detector to uncover the false data injection attacks, as shown in Fig. 5.11. From Fig. 5.7, we can see that tracking a connected vehicle based on the IMM filter is highly maneuverable while predicting the future location of the connected vehicle, and it outperforms the single filter model. The single constant acceleration filter model has the more significant

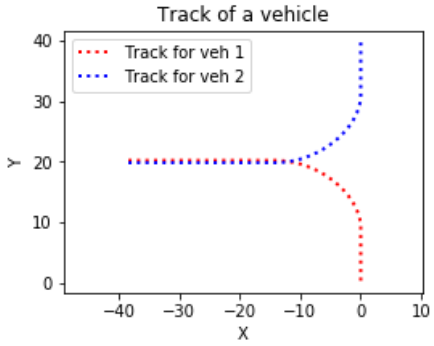


Figure 5.8: Two trajectories samples where the blue one for a normal vehicle and the red one is produced by the attacker

position and rate errors, and it is slow to recover non-maneuvering error levels after the maneuver ends. Even though the IMM tracker in the state estimator phase has superior performance in detecting trajectories anomalies than a single model tracker, it cannot catch all kinds of attacks. For example, two vehicles in an intersection can enter the same lane or road and collide if they have replayed trajectories messages or produced stealthy attack messages, thus fooling the IMM tracker since the RMS errors will be minor. Therefore, the collision identifier phase uses Time-to-Collision(TTC) metric as an action that can assist in detecting such scenarios, as shown in Fig. 5.8.

To build an efficient collision identifier, we generated all possible scenarios that an ego vehicle can interact with other remote CVs, which are defined as different RL states as described in Section 5.3. These RL states were utilized in the learning process to update and finalize the reinforcement learning Q-table in the collision identifier so that CVGuard uses collision identifier at each time step to detect *Replay* or *Stealthy* attacks. *Stealthy* attack is challenging to be detected since it is based on the physics of the vehicular system, and it is used to maximize the damage to the system while avoiding detection. Thus, we consider the worst-case scenario for CVGuard in which an attacker is undetected while injecting falsified information into the system continuously. For example, Fig. 5.12 shows that the next state estimator filter failed to

detect any malicious activity due to the normal values of the measured position prediction errors. However, the collision identifier was able to detect this attack through observing the low values of "Q-value" of the RL algorithm, which represent the action outputs of a connected vehicle's states during this attack, as shown in Fig. 5.13.

Moreover, we tested CVGuard against application-level attacks presented in [17]. These attacks include: 1) merging over large distances attack where the attacker is located

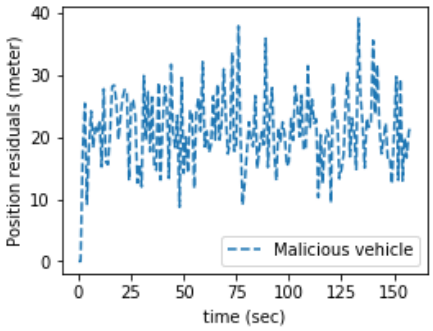


Figure 5.9: Measured position residuals for a malicious vehicle

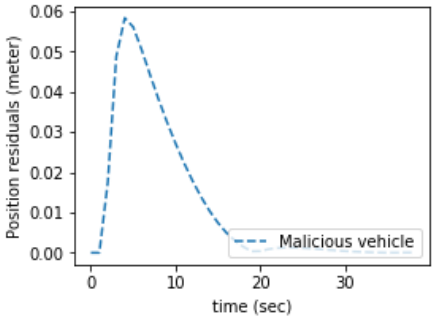


Figure 5.10: Measured position residuals for another malicious vehicle

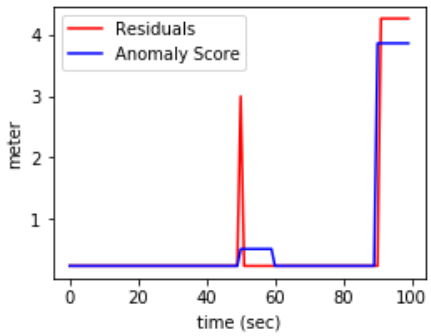


Figure 5.11: The detection statistic of the change detector component in CVGuard

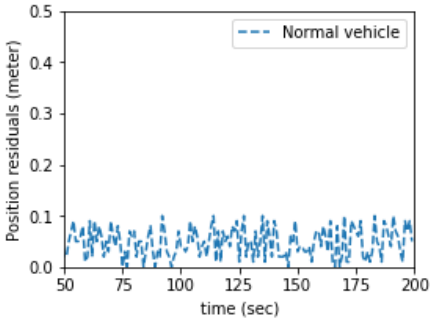


Figure 5.12: Measured position residuals (stealthy attack)

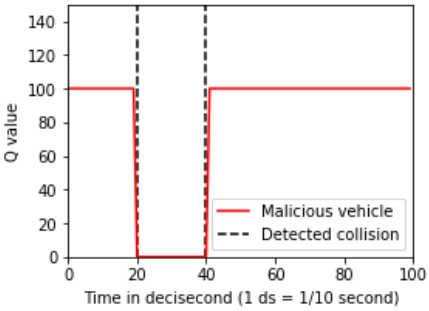


Figure 5.13: The Q-value of the reinforcement learning algorithm used in the collision detector component

between two platoons such that it can communicate with both platoons simultaneously and deceive ranging sensor by pretending that it is a member of the front platoon; 2) merging across different lanes attack where a malicious vehicle is in front of the rear platoon and sends messages pretending to be a part of the other platoon (in another lane); 3) platoon takeover attack where an attacker transmits fake messages of a fake platoon so that the rear platoon merges with the attacker. Thus, this platoon becomes under the attackers' control and can be manipulated dangerously. As shown in Fig. **??**, the space gap values are too large during the attack so that the logic identifier component in CVGuard is able to detect such attacks.

Finally, we conducted a series of experiments to compare our CVGuard accuracy with the other anomaly detector such as in [27] in terms of false positive and true positive rates. In general, the accuracy of the classifier is critical to make a security-based decision since the longer we wait, the less valuable an anomaly alert will be. The anomaly detector in [27] targets CVs'

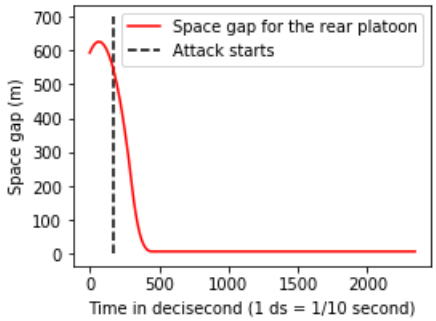


Figure 5.14: Gap between two platoons under Merge over distances attack

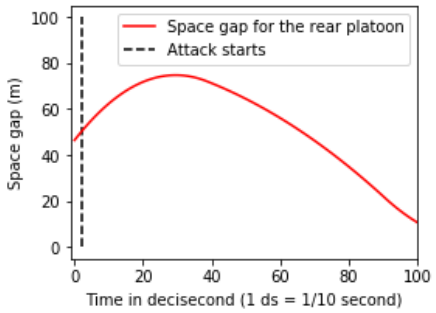


Figure 5.15: Gap between two platoons under Platoon takeover

security only in CACC applications, and it is based on machine learning. Fig. 5.16 shows that the Receiver Operating Characteristic (ROC) curve for our CVGuard detection system outperforms the other scheme by 3%. In particular, CVGuard can detect attacks with a probability close to 1 while having a low false alarm rate (less than 3%). In practice, we can lower the false positive rate by requiring multiple anomalous detection before raising an alarm, although this could delay detection.

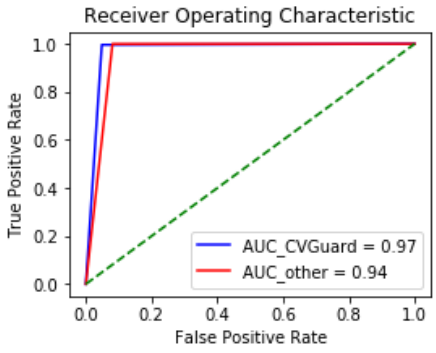


Figure 5.16: ROC curve for detection strategy.

Chapter 6

Secure Ramp Merging using Blockchain

6.1 Background

Blockchain (BC) technology is a distributed ledger technology, enabling participants of the system to agree on a transaction and log it in an unforgeable shared ledger that can be used as a record of the agreement. We propose to leverage BC to support managing and maintaining historical transactions in a Connected Vehicle (CV) environment. This allows any node in the system (i.e., vehicle or roadside infrastructure) to access past event list and its related information in the blockchain, and use that for example, to establish trust in vehicles based on past behavior. In our scheme, we use BC to ensure data immutability and automated information exchanges between different trusted nodes in a safe manner. Moreover, we rely on a credit based consensus protocol which can be seen as a credit score system to estimate the trust level in a vehicle: the higher the node's credit score is, the higher the trust level of this node would be. Using these credit scores, it is possible to separate trusted from untrusted nodes, and take that information

into account in critical maneuvers in the system. This ability is important for cooperative CV applications such as ramp merging [126] and intersection management [87], since they are time critical maneuvers for road safety and traffic efficiency where a malicious participant can substantially interfere with the system.

6.1.1 Consensus Protocols

BC [67] is a distributed ledger spreading across nodes which can be used to verify transactions on a P2P network. This is the key feature of BC that enables its unique decentralized property. It is important for BC to ensure agreement on which information is added or discarded. These processes or rules, are essentially known as a *consensus protocols* in the distributed computing community which ensure that a group of participants can reach consensus on a value even in the presence of malicious participants. Consensus is used to verify transactions and help keep the network safe.

A consensus protocol needs to be set up before the blockchain is created and it is the heart of a BC network. It provides a method of reviewing and confirming what data should be added to the blockchain's record. Because a BC network typically has no centralized authority to oversee consensus, all nodes on a BC must agree on the state of the network, following the predefined rules or protocols. Many consensus protocols have been introduced in BC technology, such as *Proof of Work*, *Proof of Stake*, *Proof of Time*, *Proof of Authority*, etc. However, we will focus on the two most widely used protocols in this paper, i.e., *Proof of Work*, and *Proof of Stake*.

Proof of Work (PoW) [125] is the first consensus protocol for BC that allows the participants to reach consensus in Bitcoin. The protocol is primarily based on a costly computation involving Hashing (SHA-256), Merkle Tree and P2P networking for creating, broadcasting and verifying blocks in the network [125]. PoW introduces the concept of mining which involves validation of a set of transactions (block) in the network by means of showing computational

proof of the completed work. When a transaction is initiated, all the miners on the network race against each other to be the first to solve a cryptographic puzzle and to create the block. The miner who successfully solves the puzzle, will broadcast his/her solution (in the form of the block) over the network to other peers. After verification of the solution, the new block will be created and accepted on the chain. Proof of work is a protocol that has the main goal of deterring cyber attacks such as distributed denial-of-service (DDoS) attack which has the purpose of exhausting the resources of a computer system by sending multiple fake requests since there is a cost (mining) to each valid request.

Proof of Stake (PoS) [125] is another consensus protocol that chooses the validator to mine the next block on the basis of its stake in the network (amount of coins a validator owns) and the age of that stake. PoS comes in many variants from minimal to significant changes in their base protocol. The most important difference among the variants is the strategy each implements to minimize the double spending and centralization issue in the protocol. Under PoS, the attacker needs to obtain 51% of participating nodes to carry out a 51% attack. Unlike PoW, an attacker in a PoS system is highly discouraged from launching 51% attack because the attacker would have to risk of loss of the entire stake if they are determined to have acted maliciously. The PoS based ledger keeps track of all the validators (equivalent to miners in PoW) and their respective stake (cryptocurrency) in the network. In PoS, all the validators invest stake in the system to earn the chances to mine the next block: the higher the stake, the higher the chances. However, it does not guarantee that the validator with the highest stake will be selected. The system chooses the validator stochastically for block creation holding a lottery with the validators with higher stakes having a higher chance of winning. If a participant tries to cheat the system, they lose their stake in the system. In our work, we use PoS because it does not require any significant computational power and provides a safer network due to the overwhelming attack costs (since the attacker has to acquire 51% of a network's stake tokens).

6.1.2 Blockchain Application for Connected Vehicles

Blockchain (BC) is an exciting and versatile technology that has been studied in different application domains. However, few studies have explored using BC in a Connected Vehicle (CV) environment. In this section, we highlight some studies on the deployment of efficient incentive mechanisms and privacy-preservation based on BC technology for CVs. Li et al. [69] introduced CreditCoin, an incentive mechanism aiming to improve robustness of crowd-sourcing systems while preserving users' privacy. In this system, when a vehicle detects an abnormal situation, it asks surrounding vehicles to confirm certain information about this abnormal event. Once validated, this information is transmitted to distant vehicles, allowing them to adjust their behaviors based on current road conditions. If a vehicle planned to acquire traffic conditions in certain area, it would provide a reward for any information transmitted by vehicles located in this area. Their proposed system was composed of three main parts: announcement protocol, privacy-preservation and incentive mechanism. To deal with information verification, they considered that designing a new BC ledger was required. However, the implementation complexity, overheads, and security properties of this system are unclear. Singh and Kim [107, 106] explore trust establishment and incentive mechanisms for CV using BC. The objective is to define a framework enabling secure communications in the CV environment without a central authority. They present a new BC ledger that allowed each vehicle to generate a unique identifier called *Bit Trust*. Moreover, BC was used to store the communication history of each vehicle. To get a reward and improve its Bit Trust, a vehicle should contribute to the proper functioning of the network. For example, if a vehicle was involved in the management of an intersection, it would get a reward and increase its Trust Bit by computing the crossing order of this intersection. However,whether their system could guarantee the security of the information exchanges using this BC technology was not fully explored.

Our work addresses challenges in applying BC consensus mechanisms, such as PoS protocol, to sustain and securely distribute trustworthy scores of CVs. We use a novel design that utilizes a data-driven methodology to detect malicious behavior with decentralized secure infrastructure to track it.

6.2 Threat Model

The threat model describes our assumptions on the attacker and their capabilities. We consider malicious vehicles that can generate falsified messages and broadcast them to other vehicles. These vehicles are *insider attackers* with previously obtained valid authentication from the *Security Credential Management System* (SCMS)[30]. SCMS is a message security solution for vehicle-to-vehicle (V2V) and vehicle-to-infrastructure (V2I) communications. It uses a Public Key Infrastructure (PKI)-based approach that employs highly innovative methods of encryption and certificate management to facilitate trusted communication. SCMS utilizes short term security credentials known as *pseudonyms* generated and updated by each vehicle to reduce its trackability (i.e., to make it difficult for eavesdroppers to tell whether BSMs transmitted at the two distinct locations are originated from the same vehicle).

Our attack model considers that an insider may have access to a vehicle with an on-board unit (OBU). This malicious vehicle is assumed to have the required credentials like a legitimate user, actively participating and sending fake data [18]. In addition, the attacker is assumed to have the capability to modify any fields in the BSM elements but not to spoof the identities in the messages since this is prevented by the SCMS certificates. In a message spoofing attack, the attacker can send out falsified position and velocity data of itself, which may induce the victim vehicles to accelerate or decelerate. This may degrade the traffic efficiency and even put vehicles near the on-ramp at risk of collisions [18]. The attacker is assumed to have the

capability of controlling the transmission rate of its OBU using a malicious unit/software. Further, the attacker is supposed to have the capability to use desired pseudonyms certificates since it can manage its own OBU. We consider only these network based attacks; we do not consider other attacks such as sensor manipulation attacks or physical attacks.

6.3 Proposed System Architecture

Our proposed scheme includes three phases: *system initialization*, *trust value calculation*, and *distributed ledger construction and maintenance*, as shown in Figure 6.1. The first phase represents the stage that the connected vehicles gets enrolled and obtains certificates from the Security Credential Management System (SCMS). The second phase is the trust value calculation for each vehicle to measure its reliability. And the last phase refers to the distributed ledger which is shared and consistent via consensus and synchronized to show the recorded vehicular transaction data. The details of all phases are presented in the remainder of this section.

6.3.1 System Initialization

A connected vehicle has to obtain a valid certificate before participating in the system. The certificate binds the owner's identity to a pair of encryption keys (i.e., public and private) which are used to encrypt and sign information. Only nodes with valid security certificates and credentials are able to send authenticated messages that will be trusted by the receiving nodes, and participate and contribute to any platform used by the CV system. To obtain certificates, a

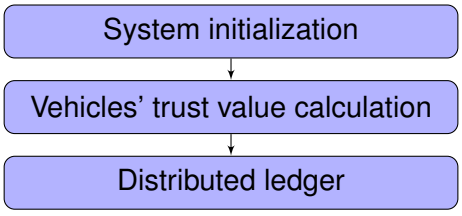


Figure 6.1: Flowchart of the proposed system

vehicle has to get enrolled into SCMS by submitting an enrollment request to U.S. Department of Transportation. Once a vehicle is authenticated, it can obtain information such as *Vehicle Trust Values* about other vehicles in the same region through the distributed ledger. Vehicle trust value of the requesting vehicle is updated continuously and can be shared among authenticated entities within the system.

6.3.2 Vehicle Trust Value Calculation

This phase is needed to create the vehicle trust estimates based on a data-driven approach to identifying falsified vehicular data. The overview of this phase is shown in Fig.6.2. This method includes *trajectory acquisition*, *feature extraction*, and *abnormal behavior determination* which is based on an artificial neural network (ANN) model and hierarchical clustering. The details of each step are presented next.

KwInputsInput KwOutputsOutput

FnCreate_Training_Table Function

Trajectory collection

In our scheme, CV applications mainly rely on basic safety messages (BSMs) which contain dynamic information such as vehicle position, speed, time stamp, acceleration, and other state variables. A trajectory is composed of multiple data instant that reveal information about

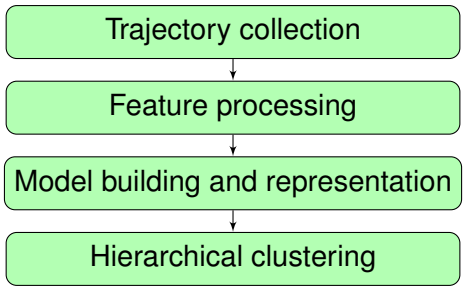


Figure 6.2: Flowchart of vehicle trust value calculation

Algorithm 5 Creating Features Extraction Table for ANN Model

▷ We digitize the following parameters for each trajectory into discrete values:
$acceleration\_rate \ \ or(A) = \{0,5,10,\dots\}$
$location\_value \ \ or(L) = \{1,2,3,\dots\}$
$range\_difference \ \ or(R) = \{0,1,2,\dots\}$
$Anomaly\_value or \ \ (AV) = \{0,1,2,\dots,10\}$

$training\_set$
S, L, R
$Features \leftarrow []$
$Index \leftarrow 0$
$N1 \leftarrow length(S)$
$N2 \leftarrow length(L)$
$N3 \leftarrow length(R)$

for $s_i \leftarrow 0$ to $N1$ **do**

    for $l_i \leftarrow 1$ to $N2$ **do**

        for $r_i \leftarrow 0$ to $N3$ **do**
$Features[\text{Index}] \leftarrow s_i + l_i + r_i + AV[\text{Index}]$
$Index \leftarrow Index + 1$
$training\_set \leftarrow Features$

path behavior over time. Different trajectories are reported by CVs through vehicle-to-vehicle (V2V) or vehicle-to-infrastructure (V2I) communications, to provide richer spatial and temporal information for better traffic management assessment. Each trajectory data contains BSM data such as location and speed of the vehicle. The selected validator is responsible for using these trajectories to process the Vehicle Trust Value calculation for each vehicle later on. If a vehicle manipulates its information, the Vehicle Trust Value Calculation algorithm will label it with a low trust score which leads to disallowing it from participating in maneuvers.

Feature Extraction

Feature extraction is used here to obtain key information from the collected trajectory data for identifying certain patterns that indicate abnormality. In this study, we use three parameters or features that can differentiate various trajectories and help putting them into

distinct clusters. These parameters are: a) acceleration rate, b) location index, and c) deferential range. The acceleration rate is a change in velocity for a vehicle through two consecutive time instances and is calculated by dividing vehicle locations by a time difference of two time instances. The location index includes any static position information such as road number, lane number, etc. Finally, deferential range is the space gap between a vehicle and its front vehicle minus the distance measured by the radar sensor. These parameters are inferred using BSMs data to represent each trajectory to be used later on.

In our scheme, the three features are then mapped into discrete ranges, such that defining these trajectories is less computationally demanding and easier to categorize, as shown in algorithm. 5. The trajectory features are fed into the neural network model. The output of the neural network model is a value that will be used in the clustering algorithm to represent a cluster later on.

Artificial Neural Network Model

In the Artificial neural network (ANN) model, information from the input neurons are multiplied by individual weights at the entry and fed into the body of the artificial neurons where these weighted inputs are summed up with biases, and processed and passed through transfer function to output. The artificial neuron can be mathematically modeled as follows:

$$y(k) = f(\sum_{n=0}^{m} w_i(k) * x_i(k) + b_i)$$

where $x_i(k)$ is the input value in the discrete time k and i ranging from 0 to m; $w_i(k)$ is the weight value in the discrete time k; b_i is the bias; f is the transfer function; and $y(k)$ is the output value in the discrete time k. Our goal is to train the ANN model by having the suitable weights for the hidden layers, so that it can generate the right output based on the right trajectory features.

We use an ANN model with one input layer (three neurons one for each trajectory feature), two hidden layers (64 neurons each) and an output layer that includes 10 neurons for probability distribution of 10 trust credit score values. In our evaluation, we use 80% of the data for training, 10% for testing, and the remaining 10% for cross validation.

Trajectory Clustering

In this phase, we apply hierarchical clustering that is an unsupervised learning algorithm that groups similar objects into clusters with similar objects. Each trajectory is treated as an element that is defined by a group of features. These features are used to compute a distance metric to identify the closest cluster to a given element. The falsified trajectory identification can be recognized through having a larger distance from existing clusters of normal trajectories. We use K-means clustering because it is a popular clustering algorithms [86]. K-means tries to partition the data set into K distinct non-overlapping subgroups (clusters) where each data point belongs to only one group. K-means assigns data points to a cluster such that the sum of the squared distance between the data points and the cluster's centroid (arithmetic mean of all the data points that belong to that cluster) is minimal. The less variation we have within clusters, the more homogeneous (similar) the data points are within the same cluster. To implement k-means algorithm, we do the following steps (as shown in Fig. 6.3): (1) we specify the number of clusters, K; (2) we initialize centroids by first shuffling the dataset and then randomly selecting K data points for the centroids without replacement; (3) we Keep iterating until there are no changes to the centroids; (4) we compute the sum of the squared distance between data points and all centroids; (5) we assign each data point to the closest cluster (centroid); and (6) we compute the centroids for the clusters by averaging all the data points which belong to each cluster.

KwInputInput KwOutputOutput PoHPoH

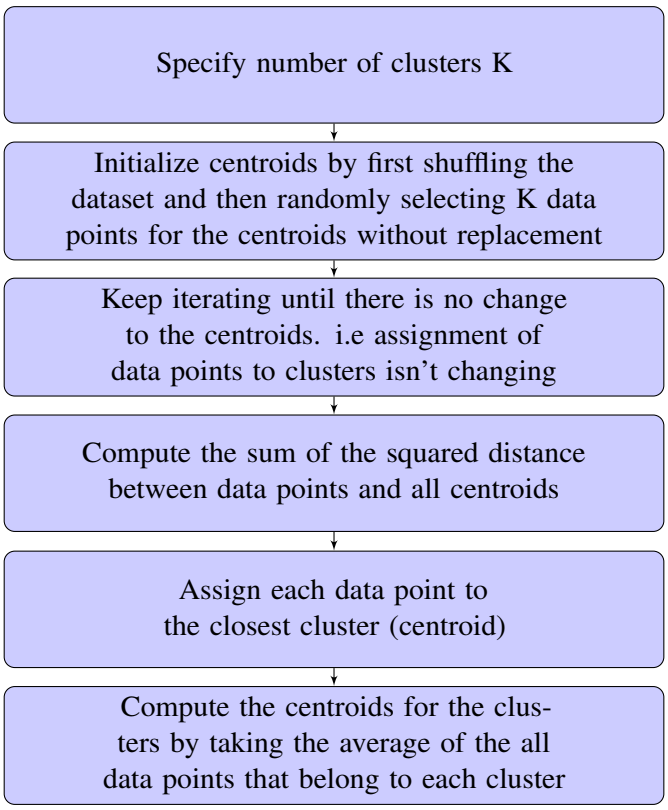


Figure 6.3: K-means clustering algorithm

Algorithm 6 Proof of History Algorithm

senderID, regionID, position Distributed block to next regions

while A CV in a regoin **do** $senderID, regionID, position$

if position = $region\_boarder_n$ **then** A change of region event is created including the updated destination trajectories for the CVs.
RSU creates and transfers PoH block to the next active regions.

6.3.3 Distributed Ledger

To increase the trustworthy of the vehicles, we propose adding long term credibility metric for the connected vehicle over time. Thus, we rely on a Proof-of-history (PoH) for verifying vehicles reliability of time between regions.

The Proof-of-history credit is mainly responsible of recording vehicles' accumulated spatial and temporal contributions into a ledger. When a vehicle moves across different regions, it is required to update its current active region. This way, the vehicles within the same active

region can communicate efficiently. The proof-of-history credit for vehicle i is computed by:

$$HS_t^i = \sum_{t=t1}^{t2} \alpha * HS_{(t2-t1)}^i \tag{6.1}$$

Where $HS_{(t2-t1)}^i$ is the proof-of-history credit during the $(t2-t1)$ period; α is a discount factor; and HS_t^i is the accumulated proof-of-history credit during the $t2$ period. This process is shown in Algorithm 6. Note that the credit point of this vehicle in the original region should be set to zero. Once vehicles' trust values are calculated, a distributed ledger can be utilized to identify and expose any abnormal behavior. This distributed ledger provides vehicle trust awareness to other surrounding vehicles, so that the vehicle can use this information before deciding to get enrolled in a certain maneuver or application. The distributed ledger is generated by selected validators that produce BC blocks. Each distributed ledger records vehicles' transaction information such as transaction ID(TID), transaction type(TT), sender ID, credit range, and region ID. In addition, timestamp will be added automatically for each record in the header, which makes it traceable. Then, the distributed ledger validation sender encrypts it with its private key and broadcasts it. To distribute ledger to other vehicular nodes, validator's election for block generation has to be performed, whose process is shown in Fig.6.4. Firstly, a vehicle credit is calculated through the equation:

$$vehicle\_credit = HS^i + TS^i + VS^i \tag{6.2}$$

where HS^i or PoH credit is calculated based on previous credits for the vehicle and can be shared by all the RSU nodes; TS^i or trust score is calculated by the *Vehicle Trust Value Calculation* algorithm; and VS^i or validation score is estimated by measuring the vehicle through different sensors. Then, each vehicle gets a credit range based on its credit value. For example, if the vehicle has a high credit, it gets a range value of 10. If the vehicle has a low credit, it gets a range

value as 0. Then, the credit range values of some random vehicles will be collected together in a group or pool. However, this pool includes more vehicles with a high credit range value and less number of vehicles with a lower credit range values which is similar to the concept of POS validator election process. Moreover, a pseudo-random election process will be used to select a validator based on a combination of factors such as the staking age, randomization, and the node's credit range value. Next, the process continues updating validator pool and selecting a validator. A validator has to be elected periodically to manage updating the blockchain due to the decentralized structure of BC technology. The election of a validator ensures the update of data in BC in a timely manner. Finally, the selected validator will be responsible for creating the distributed ledger and broadcasting it.

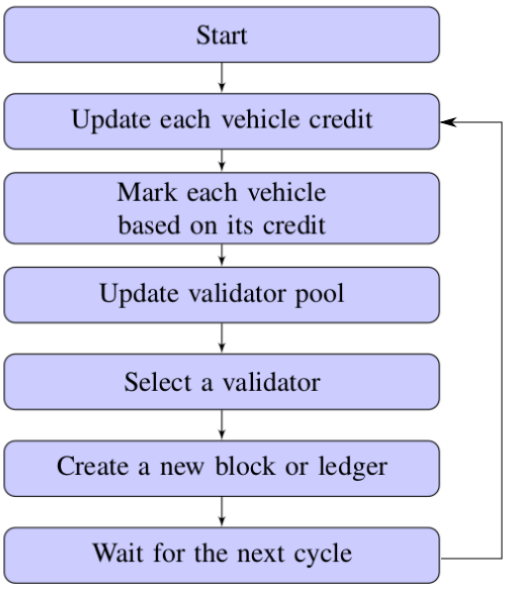


Figure 6.4: Validator election process

6.4 Cooperative Ramp Merging Algorithm

In this study, we use a cooperative ramp merging application (see Fig. 6.5) to illustrate the proposed blockchain (BC) technology. In this application, a target connected vehicle (CV)

can merge with other CVs from the mainline safely and smoothly through V2X communications. A feedforward/feedback motion control algorithm is developed to obtain the recommended longitudinal acceleration, a_{ref}, [129], which takes into account the target vehicle length l, longitudinal position r, longitudinal speed v, longitudinal acceleration a, and dynamic states from the involved remote vehicles.

$$a_{ref}(t+\delta t) = -\alpha_{ij}k_{ij}\cdot\Big[\Big(r_i(t) - r_j\big(t-\tau_{ij}(t)\big) + l_j + v_i(t) \cdot \big(t_{ij}^g(t)+\tau_{ij}(t)\big)\Big) + \gamma_i\cdot\Big(v_i(t) - v_j\big(t-\tau_{ij}(t)\big)\Big)\Big] \quad (6.3)$$

where α_{ij} denotes the value of adjacency matrix; k_{ij} and γ_i are control gains, respectively; $\tau_{ij}(t)$ denotes the time-varying communication delay between two vehicles; and $t_{ij}^g(t)$ is the time-varying desired time gap between two vehicles. Therefore, the recommended speed can be computed as:

$$v_i(t+\delta t) = v_i(t) + a_{ref}(t+\delta t)\cdot\delta t \quad (6.4)$$

where $v_i(t+\delta t)$ is the suggested speed; $v_i(t)$ is the current speed of the vehicle; and δt is the length of each time step.

6.5 Evaluation

For our experiments, we use VEhicular NeTwork Open Simulator (VENTOS)[22], a closed-loop VANET simulator that combines the capabilities of both communication network

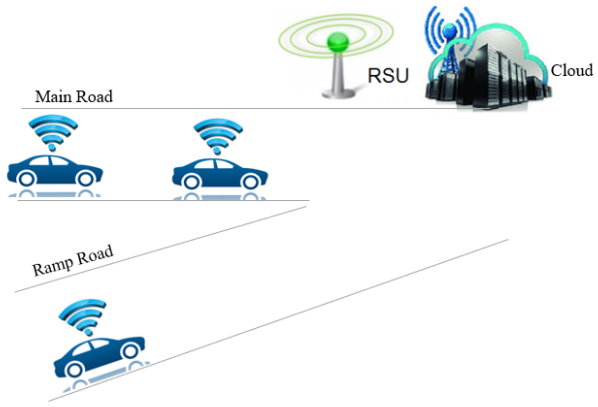


Figure 6.5: Cooperative ramp merging scenario

and vehicular traffic simulators. It is a free and open-source simulator which is designed for traffic flow analysis and intelligent traffic control, collaborative automated driving, etc. VENTOS allows for Vehicle-to-everything (V2X) communication via dedicated short-range communication (DSRC) or other means. In the simulation, vehicles are generated with Poisson distribution and spawn into a 3-mile network consisting of a 3-lane mainline segment and a single lane on-ramp. We run CVs equipped with DSRC at a maximum speed of 70 mph. The communication range for each vehicle is 300 meters and the roadside units (RSU) is located at the lane merging area. We develop our blockchain (BC) scheme including Transactions to Proof of Stake Consensus in a P2P Network of Nodes in Python as shown in Fig.6.6.

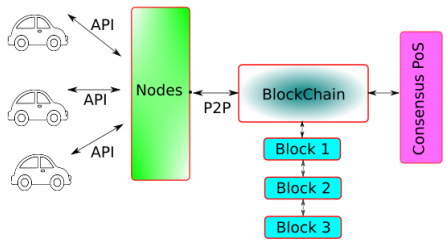


Figure 6.6: The proposed blockchain architecture

In our BC scheme, nodes/vehicles use representational state transfer(REST) API to programmatically query and invoke transactions, and to manage BC network. Our scheme has an

Account Balance Model to keep track of the balance of each account as a global state. Fig.6.7 shows the responding block size based on the number of vehicles in a CV region.

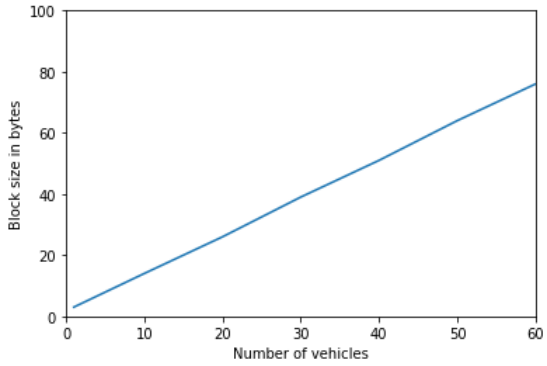


Figure 6.7: Block size vs. the number of vehicles.

To show the effects of our attacks, we apply different spoofing attacks to influence the mainline traffic. We measure the total traffic flow for the mainline as shown in Fig.6.8.

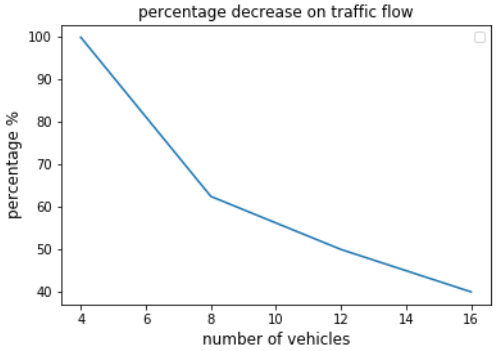


Figure 6.8: Attack impact on cooperative ramp merging

While developing our mitigation scheme, we need to determine the optimal values of system parameters such as cluster number. Thus, we sweep the values of k from 1 to 30. For each k, we calculate the total within-cluster sum of square (wss). Then, we plot the curve of wss according to the number of clusters, k. The location of a bend (knee) in the plot is generally considered as an indicator of the appropriate number of clusters as shown in Fig.6.9.

Then, we use our simulation to generate normal trajectories based on Newell's car-following model and falsified trajectories to achieve the attacker's goal. Figure 6.10 shows that

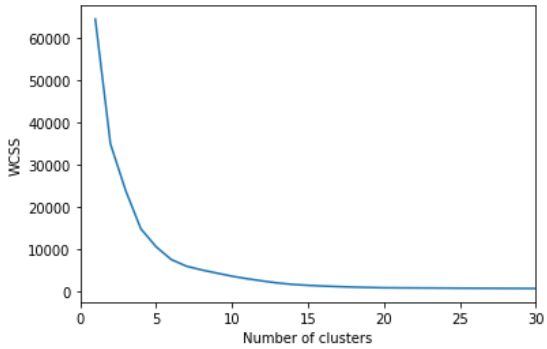


Figure 6.9: Elbow method

the distance between the cluster of falsified trajectory and clusters of other normal trajectories is so significant. This indicates that the proposed clustering method can well identify the falsified trajectory.

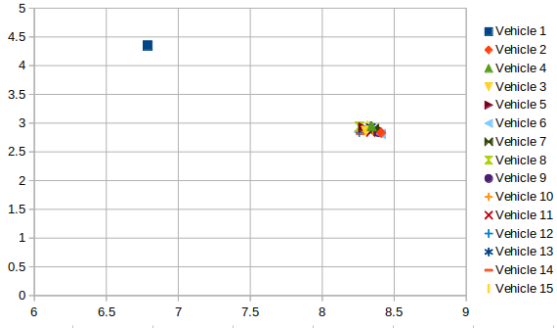


Figure 6.10: Clusters' representation for 15 trajectories

To evaluate efficiency, we measure the execution time for the Vehicle Trust Value Calculation method in our Cooperative Ramp Merging scheme. The results show that this method does not exceed 0.025 seconds as shown in Figure. 6.11, which indicates the real-time applicability of the proposed method.

Figure 6.12 shows the scenario where traffic in the on-ramp margin is under attack. Around the time instant of 30 seconds, the attacker starts its spoofing attacks. If we assume that the forger is selected to start creating the transaction block within the region after one second, then this block will be produced and distributed in less than 2 seconds. To our best knowledge, this is by far the quickest process compared to other purposed BC technology in intelligent

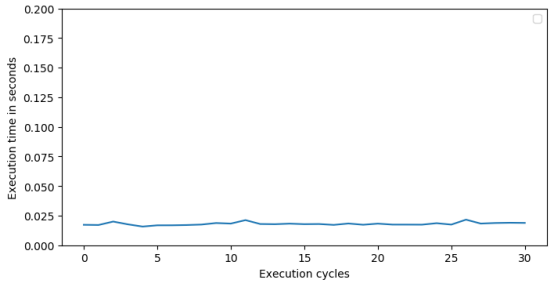


Figure 6.11: Time evaluation for vehicle trust evaluation calculation method

transportation system applications. Therefore, the design of our framework ensures that the attack can be detected immediately and the system can return to the normal condition shortly.

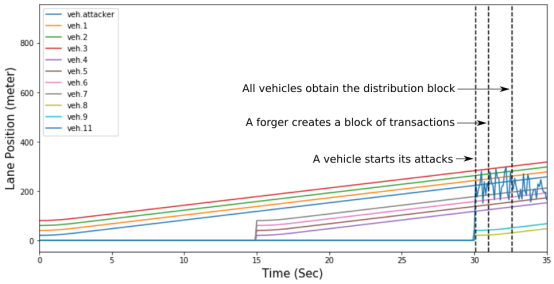


Figure 6.12: Effectiveness of our framework against injected attack in the cooperative ramp merging application.

Comparatively, in the scenario of message spoofing attack without the proposed framework, the merging vehicle on ramp will be fooled to speed up so that it creates congestion causing other mainline vehicles to decelerate. This results in degradation of over traffic performance as shown in Table 6.1. We compare both average speed and CO emissions of the merging vehicles under three different cases, i.e., without attack, under attack, and with our framework. The results show that without the protection from our framework, the attack can lead to a 45.3% decrease in average speed and an 21.3% increase in CO emission. Our proposed scheme is able to significantly improve resilience of the system.

Performance	without attacks	with attacks	Using Our scheme
average speed (m/s)	6.91	3.78	6.05
CO (mg)	43.66	52.94	43.0

Table 6.1: Economic evaluation of our framework against spoofing attacks.

Chapter 7

Concluding Remarks

To conclude, I show that Connected Vehicles (CVs) applications are a promising subject in the field of smart transportation, which is garnering interest from the USDOT, because they give rise to a new era where vehicles and transportation infrastructure are all interconnected wirelessly. However, many applications are still not considering their security vulnerabilities. I show that one of the most complete reference implementations of a CVs protocol (for Cooperative Automatic Cruise Control) is vulnerable to attacks of many types, even under a threat model that considers the state-of-the-art SCMS certificate-based security standard being developed for these applications. The attacks that exploit the vulnerabilities of these communication protocols may lead to a complete reversal of the benefits made by CVs, and as such, they have further to go before being reliably safe from attacks. I demonstrate these attacks in simulation and showed their impact on safety, performance, and economy of the traffic. Then, I present a new comprehensive framework for detecting different attacks against CVs based on state estimation, maneuvers monitoring, and reinforcement learning techniques. My evaluation of the mitigation framework showed that my mitigation attempts can diminish the introduced attacks, making it a promising approach to support the system resilience of CV applications.

Moreover, I propose a new comprehensive framework for protecting autonomous vehicles from attacks that manipulate sensor data, based on monitoring of control invariants for the vehicles. I identify several shortcomings in recent defenses that rely on using predictions of the state of the model to detect anomalies as a deviation from predictions and sensor data. In particular, I show that there is a need for effectively configuring the EKF filter parameters to enable it to more accurately model the dynamics of the system. In addition, I show that limitations in the stepwise linear model lead to substantial errors in highly dynamic phases of operation and do not account for external disturbances and noise. I propose a machine- learning, residual estimation module to compensate for these effects. Finally, I use a change aware model to more accurately detect deviations in the predicted data. Taken together, the solution substantially improved the accuracy of the prediction, leading to substantially higher detection performance with fewer false positives. I evaluate the scheme in both ground AVs and a quadcopter, using both simulation and hardware testbeds, demonstrating the effectiveness and practicality of the solution. I believe my defense makes a significant step in defending against these important attacks, and believe that the implications are systematic and generalizable.

For future work, I hope that the lessons learned from these directions help us to extend my defense framework to multi-agent connected drones. Moreover, I will investigate designing a secure recovery mechanism or technique to maintain the targeted vehicles against such attacks and take action once an attack is detected to protect the people around it. I believe this would assure better performance for CVs environment.

Bibliography

[1] CV Pilot Deployment Program. Accessed from https://www.its.dot.gov/pilots/cv_pilot_apps.htm.

[2] OMNeT++. Accessed from https://www.omnetpp.org/.

[3] PeMS - Caltrans Performance Measurement System. Accessed Aug 2018 from http://pems.dot.ca.gov/.

[4] SUMO - Simulation of Urban Mobility. Accessed from http://sumo.dlr.de/index.html.

[5] USDOT: Security Credential Management System (SCMS). Accessed from https://www.its.dot.gov/factsheets/pdf/CV_SCMS.pdf.

[6] VEINS - Vehicles in Network Simulation. Accessed from http://veins.car2x.org/.

[7] VENTOS - VEhicular NeTwork Open Simulator. Accessed from http://maniam.github.io/VENTOS/.

[8] Connected Vehicle Pilot Deployment Program, Wyoming, 2018. Accessed August 2018 from https://www.its.dot.gov/pilots/pilots_wydot.htm.

[9] Connected Vehicle Pilot Project, New York City, 2018. Accessed August 2018 from http://www.cvp.nyc.

[10] Connected Vehicle Pilot Project, Tempa, 2018. Accessed August 2018 from https://www.tampacvpilot.com/.

[11] The Open Source Application Development Portal for Federal Highway Administration, USDOT, 2018. Accessed Aug 2018 from https://www.itsforge.net/index.php/community/explore-applications/for-search-results#/30/63.

[12] Pixhawk 4, 2020. Accessed 2021 from http://https://docs.px4.io/v1.9.0/en/flight_controller/pixhawk4.html/.

[13] QGroundControl, 2020. Accessed 2021 from http://qgroundcontrol.com/.

[14] TensorFlow is an end-to-end open source platform for machine learning, 2020. Accessed 2021 from https://www.tensorflow.org.

[15] Automated Vehicles for Safety, 2021. Accessed 2017 from https://www.nhtsa.gov/technology-innovation/automated-vehicles.

[16] Genetic Algorithm, 2021. Accessed 2017 from https://www.mathworks.com/discovery/genetic-algorithm.html.

[17] ABDO, A., MALEK, S. M. B., QIAN, Z., ZHU, Q., BARTH, M., AND ABU-GHAZALEH, N. Application level attacks on connected vehicle protocols. In *22nd International Symposium on Research in Attacks, Intrusions and Defenses (RAID 2019)*, USENIX Association.

[18] ABDO, A., MALEK, S. M. B., QIAN, Z., ZHU, Q., BARTH, M., AND ABU-GHAZALEH, N. Application level attacks on connected vehicle protocols. In *22nd International Symposium on Research in Attacks, Intrusions and Defenses ({RAID} 2019)* (2019), pp. 459–471.

[19] ABDULLAH, M., JAMIL, J., AND E. MOHAN, A. M.a. abdullah 2016 - vehicle dynamics modeling simulation 2.pdf, 02 2020.

[20] AGUERO, C., KOENIG, N., CHEN, I., BOYER, H., PETERS, S., HSU, J., GERKEY, B., PAEPCKE, S., RIVERO, J., MANZO, J., KROTKOV, E., AND PRATT, G. Inside the virtual robotics challenge: Simulating real-time robotic disaster response. *Automation Science and Engineering, IEEE Transactions on 12*, 2 (April 2015), 494–506.

[21] AMINIKHANGHAHI, S., AND COOK, D. A survey of methods for time series change point detection. *Knowledge and Information Systems 51* (05 2017).

[22] AMOOZADEH, M., CHING, B., CHUAH, C.-N., AND GHOSAL, D. Ventos: Vehicular network open simulator with hardware-in-the-loop support. *Procedia Computer Science 151* (01 2019), 61–68.

[23] AMOOZADEH, M., DENG, H., CHUAH, C.-N., ZHANG, H. M., AND GHOSAL, D. Platoon management with cooperative adaptive cruise control enabled by vanet. *Vehicular communications 2*, 2 (2015), 110–123.

[24] AMOOZADEH, M., RAGHURAMU, A., N. CHUAH, C., GHOSAL, D., ZHANG, H. M., ROWE, J., AND LEVITT, K. Security vulnerabilities of connected vehicle streams and their impact on cooperative driving. *IEEE Communications Magazine* (June 2015).

[25] BARTH, M. J., WU, G., AND BORIBOONSOMSIN, K. Intelligent transportation systems and greenhouse gas reductions. *Current Sustainable/Renewable Energy Reports 2*, 3 (2015), 90–97.

[26] BEARD, R. Quadrotor dynamics and control rev 0.1.

[27] BODDUPALLI, S., RAO, A. S., AND RAY, S. Resilient cooperative adaptive cruise control for autonomous vehicles using machine learning. *CoRR abs/2103.10533* (2021).

[28] BOIS, F. Bayesian inference. *Methods in molecular biology (Clifton, N.J.) 930* (01 2013), 597–636.

[29] BOZCAN, I., AND KAYACAN, E. Au-air: A multi-modal unmanned aerial vehicle dataset for low altitude traffic surveillance, 01 2020.

[30] BRECHT, B., THERRIAULT, D., WEIMERSKIRCH, A., WHYTE, W., KUMAR, V., HEHN, T., AND GOUDY, R. A security credential management system for v2x communications, 2018.

[31] CAO, Y., XIAO, C., CYR, B., ZHOU, Y., PARK, W., RAMPAZZI, S., CHEN, Q. A., FU, K., AND MAO, Z. M. Adversarial sensor attack on lidar-based perception in autonomous driving. *CoRR abs/1907.06826* (2019).

[32] CHEN, Q. A., YIN, Y., FENG, Y., MAO, Z. M., AND LIU, H. X. Exposing congestion attack on emerging connected vehicle based traffic signal control. In *Proceedings of the 25th Network and Distributed System Security Symposium (NDSS'18), San Diego* (2018).

[33] CHOI, H., LEE, W.-C., AAFER, Y., FEI, F., TU, Z., ZHANG, X., XU, D., AND DENG, X. Detecting attacks against robotic vehicles: A control invariant approach. In *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security* (New York, NY, USA, 2018), CCS '18, Association for Computing Machinery, p. 801–816.

[34] CUI, L., HU, J., PARK, B., AND BUJANOVIC, P. Development of a simulation platform for safety impact analysis considering vehicle dynamics, sensor errors, and communication latencies: Assessing cooperative adaptive cruise control under cyber attack. *Transportation Research Part C Emerging Technologies 97* (12 2018).

[35] DAVIDSON, D., WU, H., JELLINEK, R., SINGH, V., AND RISTENPART, T. Controlling uavs with sensor input spoofing attacks. In *10th USENIX Workshop on Offensive Technologies (WOOT 16)* (Austin, TX, Aug. 2016), USENIX Association.

[36] DEMPSEY , D. Learn the hidden secret of autonomous car navigation guidance: Imus.

[37] DOSOVITSKIY, A., ROS, G., CODEVILLA, F., LOPEZ, A., AND KOLTUN, V. CARLA: An open urban driving simulator. In *Proceedings of the 1st Annual Conference on Robot Learning* (2017), pp. 1–16.

[38] FEI, F., TU, Z., YU, R., KIM, T., ZHANG, X., XU, D., AND DENG, X. Cross-layer retrofitting of uavs against cyber-physical attacks. In *2018 IEEE International Conference on Robotics and Automation (ICRA)* (2018), pp. 550–557.

[39] FENG, Y., HEAD, L., KHOSHMAGHAM, S., AND ZAMANIPOUR, M. A real-time adaptive signal control in a connected vehicle environment. *Transportation Research Part C: Emerging Technologies 55* (01 2015).

[40] FLUIDMESH NETWORKS LLC. DSRC Roadside Unit. Accessed March 2019 from `https://www.fluidmesh.com/dsrc-roadside-unit/`.

[41] FRANK, P. M. Fault diagnosis in dynamic systems using analytical and knowledge-based redundancy—a survey and some new results. *Automatica 26*, 3 (May 1990), 459–474.

[42] FU, K., AND XU, W. Risks of trusting the physics of sensors. *Commun. ACM 61*, 2 (Jan. 2018), 20–23.

[43] GIECHASKIEL, I., AND RASMUSSEN, K. B. Sok: Taxonomy and challenges of out-of-band signal injection attacks and defenses. *CoRR abs/1901.06935* (2019).

[44] GIRALDO, J., URBINA, D., CARDENAS, A., VALENTE, J., FAISAL, M., RUTHS, J., TIPPENHAUER, N. O., SANDBERG, H., AND CANDELL, R. A survey of physics-based attack detection in cyber-physical systems. *ACM Comput. Surv. 51*, 4 (July 2018).

[45] HARDING, J., POWELL, G., YOON, R., FIKENTSCHER, J., DOYLE, C., SADE, D., LUKUC, M., SIMONS, J., WANG, J., ET AL. Vehicle-to-vehicle communications: readiness of V2V technology for application. Tech. rep., United States. National Highway Traffic Safety Administration, 2014.

[46] HARTENSTEIN, H., AND LABERTEAUX, L. A tutorial survey on vehicular ad hoc networks. *IEEE Communications magazine 46*, 6 (2008).

[47] HU, S., CHEN, Q. A., JOUNG, J., CARLAK, C., FENG, Y., MAO, Z., AND LIU, H. Cvshield: Guarding sensor data in connected vehicle with trusted execution environment. pp. 1–4.

[48] HU, S., CHEN, Q. A., JOUNG, J., CARLAK, C., FENG, Y., MAO, Z., AND LIU, H. Cvshield: Guarding sensor data in connected vehicle with trusted execution environment.

[49] HU, W. Robust stability of optimization-based state estimation under bounded disturbances.

[50] HU, Y. C., PATEL, M., SABELLA, D., SPRECHER, N., AND YOUNG, V. Mobile edge computing—a key technology towards 5g. *ETSI white paper 11*, 11 (2015), 1–16.

[51] HUANG, S., WONG, W., FENG, Y., CHEN, Q. A., MAO, Z., AND LIU, H. Impact evaluation of falsified data attacks on connected vehicle based traffic signal control, 10 2020.

[52] HUMPHREYS, T., LEDVINA, B., PSIAKI, M., O'HANLON, B., AND KINTNER, J. Assessing the spoofing threat: Development of a portable gps civilian spoofer. pp. 2314–2325.

[53] HUQ, GIBSON, K., AND VOSSELER. Cybersecurity for connected cars: Exploring risks in 5g, cloud, and other connected technologies. Trend Micro Research, pp. 1–88.

[54] ISLAM, M., CHOWDHURY, M., LI, H., AND HU, H. Cybersecurity attacks in vehicle-to-infrastructure (v2i) applications and their prevention, 2017.

[55] JAHAN, F., SUN, W., NIYAZ, Q., AND ALAM, M. Security modeling of autonomous systems: A survey. *ACM Comput. Surv. 52*, 5 (Sept. 2019).

[56] JIA, D., LU, K., AND WANG, J. On the network connectivity of platoon-based vehicular cyber-physical systems. *Transportation Research Part C: Emerging Technologies 40* (2014), 215–230.

[57] JULIER, S., AND UHLMANN, J. New extension of the kalman filter to nonlinear systems. In *Defense, Security, and Sensing* (1997).

[58] JUNKINS, J., AND SHUSTER, M. Geometry of the euler angles. *Journal of The Astronautical Sciences - J ASTRONAUT SCI 41* (10 1993), 531–543.

[59] KALMAN, R. A new approach to linear filtering and prediction problems" transaction of the asme journal of basic.

[60] KAUR, S., AND SINGH, M. Automatic attack signature generation systems: A review. *IEEE Security Privacy 11*, 6 (2013), 54–61.

[61] KENNEY, J. Dedicated short-range communications (dsrc) standards in the united states. *Proceedings of the IEEE 99* (08 2011), 1162 – 1182.

[62] KERNS, A., SHEPARD, D., BHATTI, J., AND HUMPHREYS, T. Unmanned aircraft capture and control via gps spoofing. *Journal of Field Robotics 31* (07 2014).

[63] KHATTAK, Z. H., PARK, H., HONG, S., ATTA BOATENG, R., AND SMITH, B. Investigating cybersecurity issues in active traffic management systems. *Transportation Research Record Journal of the Transportation Research Board 2672* (07 2018).

[64] KOENIG, N., AND HOWARD, A. Design and use paradigms for gazebo, an open-source multi-robot simulator. In *IEEE/RSJ International Conference on Intelligent Robots and Systems* (Sendai, Japan, Sep 2004), pp. 2149–2154.

[65] KOUBAA, A. Robot operating system (ros): The complete reference (volume 1).

[66] KRAJEWSKI, R., BOCK, J., KLOEKER, L., AND ECKSTEIN, L. The highd dataset: A drone dataset of naturalistic vehicle trajectories on german highways for validation of highly automated driving systems. In *2018 21st International Conference on Intelligent Transportation Systems (ITSC)* (2018), pp. 2118–2125.

[67] KRAMER, M. What are consensus protocols? `https://decrypt.co/resources/consensus-protocols-what-are-they-guide-how-to-explainer`.

[68] LEENSTRA, H. Multi actor road map to improve cyber security of consumer used connected cars.

[69] LI, L., LIU, J., CHENG, L., QIU, S., WANG, W., ZHANG, X., AND ZHANG, Z. Creditcoin: A privacy-preserving blockchain-based incentive announcement network for communications of smart vehicles. *IEEE Transactions on Intelligent Transportation Systems 19*, 7 (2018), 2204–2220.

[70] LI, Y., TU, Y., FAN, Q., DONG, C., AND WANG, W. Influence of cyber-attacks on longitudinal safety of connected and automated vehicles. *Accident Analysis Prevention 121* (2018), 148–156.

[71] LIN, G., MILAN, A., SHEN, C., AND REID, I. D. Refinenet: Multi-path refinement networks for high-resolution semantic segmentation. *CoRR abs/1611.06612* (2016).

[72] LIU, J., YAN, C., AND XU, W. Can you trust autonomous vehicles: Contactless attacks against sensors of self-driving vehicles. DEF CON, 2016. https://doi.org/10.5446/36252 *Lastaccessed* : 30*Jan*2021.

[73] LJUNG. The control handbook. *CRC Press* (1996).

[74] LULKA, J. Nissan leaf security flaws exposed via hacking. `https://www.digitalengineering247.com/article/nissan-leaf-security-flaws-exposed-via-hacking/`.

[75] MAZOR, E., AVERBUCH, A., BAR-SHALOM, Y., AND DAYAN, J. Interacting multiple model methods in target tracking: a survey. *IEEE Transactions on Aerospace and Electronic Systems 34*, 1 (1998), 103–123.

[76] MEIER, L., TANSKANEN, P., HENG, L., LEE, G. H., FRAUNDORFER, F., AND POLLEFEYS, M. Pixhawk: A micro aerial vehicle design for autonomous flight using onboard computer vision. *Autonomous Robots 33*, 1 (2012), 21–39.

[77] MILANÉS, V., SHLADOVER, S. E., SPRING, J., NOWAKOWSKI, C., KAWAZOE, H., AND NAKAMURA, M. Cooperative adaptive cruise control in real traffic situations. *IEEE Transactions on Intelligent Transportation Systems 15*, 1 (2014), 296–305.

[78] MINDERHOUD, M. M., AND BOVY, P. H. Extended time-to-collision measures for road traffic safety assessment. *Accident Analysis & Prevention 33*, 1 (2001), 89–97.

[79] MORRIS, T., AND GAO, W. On cyber attacks and signature based intrusion detection for modbus based industrial control systems. *Journal of Digital Forensics, Security and Law 9* (01 2014), 37–56.

[80] MURGUIA, C., AND RUTHS, J. Cusum and chi-squared attack detection of compromised sensors. In *2016 IEEE Conference on Control Applications (CCA)* (2016), pp. 474–480.

[81] MURGUIA, C., AND RUTHS, J. Cusum and chi-squared attack detection of compromised sensors. In *2016 IEEE Conference on Control Applications (CCA)* (2016), pp. 474–480.

[82] NG, A. Y. Feature selection, l1 vs. l2 regularization, and rotational invariance. In *Proceedings of the Twenty-First International Conference on Machine Learning* (New York, NY, USA, 2004), ICML '04, Association for Computing Machinery, p. 78.

[83] NGUYEN, T. T., AND REDDI, V. J. Deep reinforcement learning for cyber security. *CoRR abs/1906.05799* (2019).

[84] NOH, J., KWON, Y., SON, Y., SHIN, H., KIM, D., CHOI, J., AND KIM, Y. Tractor beam: Safe-hijacking of consumer drones with adaptive gps spoofing. *ACM Trans. Priv. Secur. 22*, 2 (Apr. 2019).

[85] OSHMAN, Y., AND SHAVIV, I. Optimal tuning of a kalman filter using genetic algorithms.

[86] PATIL, P., AND KARTHIKEYAN, A. A survey on k-means clustering for analyzing variation in data. In *Inventive Communication and Computational Technologies* (Singapore, 2020), G. Ranganathan, J. Chen, and Á. Rocha, Eds., Springer Singapore, pp. 317–323.

[87] PENG, J., HUANG, H., AND CHEN, L. An adaptive traffic signal control in a connected vehicle environment: A systematic review. *Information (Switzerland) 8* (08 2017).

[88] PEPY, R., LAMBERT, A., AND MOUNIER, H. Path planning using a dynamic vehicle model. In *2006 2nd International Conference on Information Communication Technologies* (2006), vol. 1, pp. 781–786.

[89] PERRY, F., RABOY, K., LESLIE, E., HUANG, Z., VAN DUREN, D., ET AL. Dedicated Short-Range Communications RoadSide Unit Specifications. Tech. rep., United States. Department of Transportation, April 2017.

[90] PERTIGKIOZOGLOU, S., AND MARAGOS, P. Detecting adversarial examples in convolutional neural networks. *CoRR abs/1812.03303* (2018).

[91] PETIT, J., STOTTELAAR, B., AND FEIRI, M. Remote attacks on automated vehicles sensors : Experiments on camera and lidar.

[92] PHAM, M., AND XIONG, K. A survey on security attacks and defense techniques for connected and autonomous vehicles, 2020.

[93] PLOEG, J., SCHEEPERS, B. T. M., VAN NUNEN, E., VAN DE WOUW, N., AND NIJMEIJER, H. Design and experimental evaluation of cooperative adaptive cruise control. In *2011 14th International IEEE Conference on Intelligent Transportation Systems (ITSC)* (2011).

[94] QUINONEZ, R., GIRALDO, J., SALAZAR, L., BAUMAN, E., CARDENAS, A., AND LIN, Z. SAVIOR: Securing autonomous vehicles with robust physical invariants. In *29th USENIX Security Symposium (USENIX Security 20)* (Aug. 2020), USENIX Association, pp. 895–912.

[95] QUINONEZ, R., GIRALDO, J., SALAZAR, L., BAUMAN, E., CARDENAS, A., AND LIN, Z. SAVIOR: Securing autonomous vehicles with robust physical invariants. In *29th USENIX Security Symposium (USENIX Security 20)* (Aug. 2020), USENIX Association, pp. 895–912.

[96] SAAID, A., NUR, D., AND KING, R. Change points detection of vector autoregressive model using sdvar algorithm.

[97] SABATINO, F. Quadrotor control: modeling, nonlinearcontrol design, and simulation.

[98] SEGATA, M., DRESSLER, F., LO CIGNO, R., AND GERLA, M. A simulation tool for automated platooning in mixed highway scenarios. In *Proceedings of the 18th Annual International Conference on Mobile Computing and Networking* (New York, NY, USA, 2012), Mobicom '12, ACM, pp. 389–392.

[99] SELVARAJ, J., DAYANIKLI, G. Y., GAUNKAR, N. P., WARE, D., GERDES, R. M., AND MINA, M. Electromagnetic induction attacks against embedded systems. In *Proceedings of the 2018 on Asia Conference on Computer and Communications Security* (New York, NY, USA, 2018), ASIACCS '18, Association for Computing Machinery, p. 499–510.

[100] SEN, S., AND HEAD, L. Controlled optimization of phases at an intersection. *Transportation Science 31* (02 1997), 5–17.

[101] SESHADRI, A., LUK, M., PERRIG, A., VAN DOORN, L., AND KHOSLA, P. Scuba: Secure code update by attestation in sensor networks. vol. 2006, pp. 85–94.

[102] SHIN, H., KIM, D., KWON, Y., AND KIM, Y. Illusion and dazzle: Adversarial optical channel exploits against lidars for automotive applications. In *Cryptographic Hardware and Embedded Systems – CHES 2017* (Cham, 2017), W. Fischer and N. Homma, Eds., Springer International Publishing, pp. 445–467.

[103] SHYALIKA, C. A beginners guide to q-learning. *towardsdatascience*.

[104] SIDDIQUI, A. S., GUI, Y., PLUSQUELLIC, J., AND SAQIB, F. Secure communication over canbus. In *2017 IEEE 60th International Midwest Symposium on Circuits and Systems (MWSCAS)* (2017), pp. 1264–1267.

[105] SIMPKINS, C. System identification: Theory for the user, 2nd edition (ljung, l.; 1999) [on the shelf]. *Robotics Automation Magazine, IEEE 19* (06 2012), 95–96.

[106] SINGH, M., AND KIM, S. Intelligent vehicle-trust point: Reward based intelligent vehicle communication using blockchain.

[107] SINGH, M., AND KIM, S. Introduce reward-based intelligent vehicles communication using blockchain. In *2017 International SoC Design Conference (ISOCC)* (2017), pp. 15–16.

[108] SKOGLUND, M. A., HENDEBY, G., AND AXEHILL, D. Extended kalman filter modifications based on an optimization view point. In *2015 18th International Conference on Information Fusion (Fusion)* (2015), pp. 1856–1861.

[109] SOMMER, C., GERMAN, R., AND DRESSLER, F. Bidirectionally coupled network and road traffic simulation for improved ivc analysis. *IEEE Transactions on Mobile Computing 10*, 1 (January 2011), 3–15.

[110] SOMMER, C., GERMAN, R., AND DRESSLER, F. Bidirectionally coupled network and road traffic simulation for improved ivc analysis. *IEEE Transactions on Mobile Computing 10*, 1 (January 2011), 3–15.

[111] SON, Y., SHIN, H., KIM, D., PARK, Y., NOH, J., CHOI, K., CHOI, J., AND KIM, Y. Rocking drones with intentional sound noise on gyroscopic sensors. In *24th USENIX Security Symposium (USENIX Security 15)* (Washington, D.C., Aug. 2015), USENIX Association, pp. 881–896.

[112] SRIVASTAVA, N., HINTON, G., KRIZHEVSKY, A., SUTSKEVER, I., AND SALAKHUTDINOV, R. Dropout: A simple way to prevent neural networks from overfitting. *J. Mach. Learn. Res. 15*, 1 (jan 2014), 1929–1958.

[113] STANGER, T., AND RE, L. A model predictive cooperative adaptive cruise control approach. pp. 1374–1379.

[114] STONE, M. Cross-validatory choice and assessment of statistical predictions. *Journal of the royal statistical society series b-methodological 36* (1974), 111–133.

[115] SUTTON, R. S., AND BARTO, A. G. *Reinforcement Learning: An Introduction*. A Bradford Book, Cambridge, MA, USA, 2018.

[116] SYNOPSYS, I. What is an Autonomous Car?, 2021. Accessed 2017 from `https://www.synopsys.com/automotive/what-is-autonomous-car.html`.

[117] TAYLOR, W. A. Change-point analysis : A powerful new tool for detecting changes.

[118] T.FOOTE. ROS Kinetic Kame Released, 2016. Accessed 2017 from `https://www.ros.org/news/2016/05/ros-kinetic-kame-released.html`.

[119] TING, T., MAN, K., LIM, E., AND LEACH, M. Tuning of kalman filter parameters via genetic algorithm for state-of-charge estimation in battery management system. *TheScientificWorldJournal 2014* (08 2014), 176052.

[120] TIPPENHAUER, N. O., PÖPPER, C., RASMUSSEN, K. B., AND CAPKUN, S. On the requirements for successful gps spoofing attacks. In *Proceedings of the 18th ACM Conference on Computer and Communications Security* (New York, NY, USA, 2011), CCS '11, Association for Computing Machinery, p. 75–86.

[121] TRIPPEL, T., WEISSE, O., XU, W., HONEYMAN, P., AND FU, K. Walnut: Waging doubt on the integrity of mems accelerometers with acoustic injection attacks. In *2017 IEEE European Symposium on Security and Privacy (EuroS P)* (2017), pp. 3–18.

[122] TU, Y., LIN, Z., LEE, I., AND HEI, X. Injected and delivered: Fabricating implicit control over actuation systems by spoofing inertial sensors. In *27th USENIX Security Symposium (USENIX Security 18)* (Baltimore, MD, Aug. 2018), USENIX Association, pp. 1545–1562.

[123] TYREE, Z., BRIDGES, R. A., COMBS, F. L., AND MOORE, M. R. Exploiting the shape of CAN data for in-vehicle intrusion detection. *CoRR abs/1808.10840* (2018).

[124] UMAMAGESWARI, A., IGNATIOUS, J., AND VINODHA, R. A comparitive study of kalman filter, extended kalman filter and unscented kalman filter for harmonic analysis of the non-stationary signals.

[125] WAHAB, A., AND MEHMOOD, W. Survey of consensus protocols. *CoRR abs/1810.03357* (2018).

[126] WANG, Y., E, W., TANG, W., TIAN, D., LU, G., AND YU, G. Automated on-ramp merging control algorithm based on internet-connected vehicles. *IET Intelligent Transport Systems 7*, 4 (2013), 371–379.

[127] WANG, Y., MASOUD, N., AND KHOJANDI, A. Real-time sensor anomaly detection and recovery in connected automated vehicle sensors. *IEEE Transactions on Intelligent Transportation Systems 22*, 3 (Mar 2021), 1411–1421.

[128] WANG, Z., HAN, K., KIM, B., WU, G., AND BARTH, M. Lookup table-based consensus algorithm for real-time longitudinal motion control of connected and automated vehicles, 07 2019.

[129] WANG, Z., HAN, K., KIM, B., WU, G., AND BARTH, M. J. Lookup table-based consensus algorithm for real-time longitudinal motion control of connected and automated vehicles. *arXiv:1902.07747v2* (2019).

[130] WU, Y.-J., LIAN, F.-L., AND CHANG, T.-H. Traffic monitoring and vehicle tracking using roadside cameras. *2006 IEEE International Conference on Systems, Man and Cybernetics 6* (2006), 4631–4636.

[131] WYGLINSKI, A. M., HUANG, X., PADIR, T., LAI, L., EISENBARTH, T. R., AND VENKATASUBRAMANIAN, K. Security of autonomous systems employing embedded computing and sensors. *IEEE Micro 33*, 1 (2013), 80–86.

[132] XIA, J., RAO, W., HUANG, W., AND LU, Z. Automatic Multi-Vehicle Tracking using Video Cameras: An improved CAMShift approach. *KSCE Journal of Civil Engineering 17* (09 2013), 1462–1470.

[133] ZENG, K. C., LIU, S., SHU, Y., WANG, D., LI, H., DOU, Y., WANG, G., AND YANG, Y. All your GPS are belong to us: Towards stealthy manipulation of road navigation systems. In *27th USENIX Security Symposium (USENIX Security 18)* (Baltimore, MD, Aug. 2018), USENIX Association, pp. 1527–1544.