Context-Aware Detection and Resolution of Data Anomalies for Semi-Autonomous Cyber-Physical Systems

by

Chun-Yu Chen

A dissertation submitted in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
(Computer Science and Engineering)
in The University of Michigan
2022

Doctoral Committee:

Professor Kang G. Shin, Chair
Professor J. Alex Halderman
Research Professor Peter Honeyman
Associate Professor Lionel P. Robert

Chun-Yu Chen

chunyuc@umich.edu

ORCID iD: 0000-0003-1820-1719

To my parents and my grandparents

for their unconditional love and support.

ACKNOWLEDGEMENTS

This dissertation is not only my accomplishment but also the accomplishment of all the people who have supported and helped me during my Ph.D. study.

First and foremost, I would like to thank my advisor, Prof. Kang G. Shin, not only for his guidance on my research but also for teaching me the path toward a successful, happy life. He gives me the freedom to explore my research interest, which becomes the topic of my dissertation, and helps me become a researcher that is capable of independent thinking.

I would like to thank my dissertation committee members — Prof. J. Alex Halderman, Prof. Peter Honeyman, and Prof. Lionel P. Robert — for their constructive comments and guidance to improve this dissertation. I am grateful to our research collaborators, including Yu Seung Kim, Soodeh Dadras, John Moore, Lily Yang, Kathiravetpillai Sivanesan, and Xiruo Liu, for their inputs w.r.t. constructing practical systems from industrial perspectives. I would like to thank my undergrad and master's advisor, Prof. Chun-Ting Chou, for leading me into the world of academic research and encouraging me to pursue my Ph.D. study at the University of Michigan.

I am grateful to all fellow RTCLers. I want to give special thanks to Yu-Chih, Dongyao, Liang, Duc, Kassem, Haichuan and Arun for all the help to not only my research life but also my personal life in Ann Arbor. I thank Mert, Hsun-Wei, Youssef, Taeju, Youngmoon, Juncheng, Noah, Wei-Lun, and Brian for broadening my knowledge w.r.t. different research areas and always being my good friends.

I would like to thank my parents and family for their unconditional love and support that make me who I am today.

The work in this dissertation is supported in part by the US Army Research Office under Grant No.W911NF-21-1-0057 and the National Science Foundation under Grant No. CNS-1646130.

TABLE OF CONTENTS

LIST OF FIGURES

Figure

LIST OF TABLES

Table

ABSTRACT

A cyber-physical system (CPS) with both autonomous and manual control capabilities, or a *semi-autonomous* (SA) system, is one of the most commonly seen types of system in our daily lives, such as cars, airplanes and ships. While having the benefits of autonomous control to enhance safety/comfort of transportation and the flexibility of manual control to handle safety-critical situations, SA systems inevitably inherit the vulnerabilities embedded in both control types. That is, an SA system will also suffer from component failures or design/software bugs (*e.g.*, crashes of Boeing 737 MAX) and potential attacks (*e.g.*, sensor spoofing) as a general CPS does. Moreover, since mechanical components are gradually being replaced by their electronic counterparts in SA systems, this trend also introduces new reliability and security risks — increasing adoption of multiple heterogeneous communication interfaces widens attack surfaces that an adversary can exploit.

Considering the potential security and safety concerns caused by system faults/flaws, human error, and malicious attacks, we develop a suite of mechanisms/systems for detection and resolution of system anomalies by cross-validating the sensor data and the context information to enhance the security and safety of SA systems from three key perspectives that can directly influence the operation of SA systems — system operation, received information, and control decisions. In this thesis, we propose both domain-general design for SA systems and its domain-specific realization using SA vehicles as a concrete case study. From the system operation perspective, we propose **CADD**, a context-aware anomaly detection system, to capture abnormal system behavior under various operation contexts while considering practical scenarios where some (context) information cannot be observed by the SA system. We then present **DiVa**, a diagnostic system that pinpoints/iden-

tifies the anomalous source(s) after an anomaly is detected. It exploits the cyber-physical correlation or causality between the internal data of the SA system to narrow down the origin of anomaly while assuming no data can be entirely trusted. From the received information perspective, we propose **EDRoad**, an easy-to-use system for verification of the data received from external sources to ensure no compromised data will be used to provide services to the SA systems. Finally, from the control decision perspective, we introduce **CADCA**, a system for detecting and resolving control actions that may potentially lead to unstable system states or safety-critical situations.

CHAPTER I

Introduction

While more and more autonomous functions are getting added to modern vehicles, we are now officially in the era of semi-autonomous (SA) cyber-physical systems — cyber-physical systems with both autonomous control and manual control capabilities. In this thesis, we explore and develop mechanisms and algorithms for enhancing the security and safety of SA cyber-physical systems based on detection and resolution of data anomalies.

1.1 Semi-Autonomous Cyber-Physical Systems

Semi-autonomous cyber-physical systems — simply called SA systems — are one of the most commonly seen types of systems in our daily lives, such as cars, airplanes and ships. In a typical SA system, autonomous control is responsible for common routine operations assisting human operators (e.g., drivers, pilots or helmsmen) to reduce their burden of controlling an SA system (*e.g.*, the autopilot and self-driving functions in an airplane and a car, respectively), or it is designed to respond to certain conditions to avoid/mitigate safety risks (*e.g.*, the automatic emergency braking system in a car). On the other hand, while coexisting with autonomous control, manual control provides the human operators with the ability to perform fine-grained controls that cannot yet be performed by its autonomous counterpart, and it also acts as a fail-safe design to take over the system when the autonomous control fails.

1.1.1 Typical Deployment Scenario

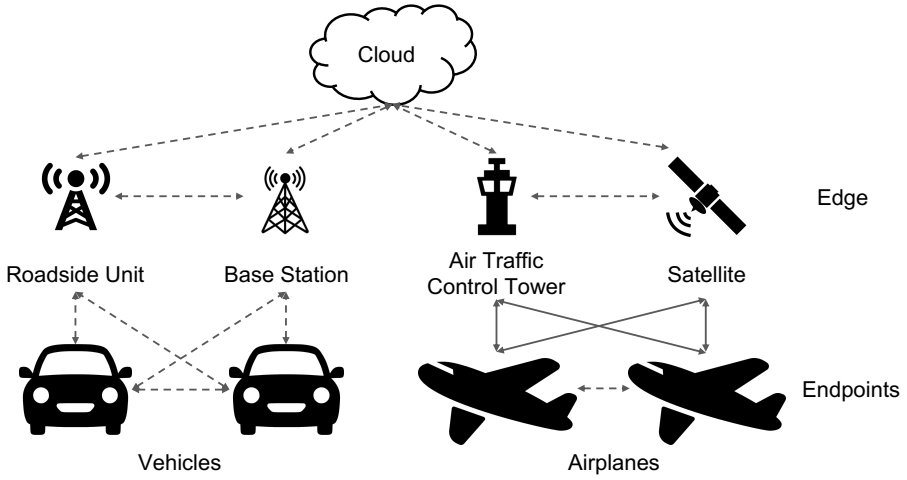


Figure 1.1: A typical deployment scenario of SA systems and the communication links between different entities, where the dashed lines indicate the links may or may not exist in real-world deployment.

Fig. 1.1 shows a typical deployment architecture in which at least three types of entities can influence the operation of an SA system — endpoints (*i.e.*, the ego SA system and other SA systems), edge, and cloud. While optional inter-entity communications may exist to assist the operation of SA systems, the three fundamental elements that can directly influence the operation of an SA system are i) the (cyber-)physical implementation of the SA system, ii) the control decisions generated by the human operator or the autonomous functions, and iii) the service provided by infrastructure (*i.e.*, edge + cloud).

1.1.2 Safety and Security Threats

While embracing the comfort and convenience brought by autonomous control and the capability of fine-grained control provided by human operators, SA systems also inevitably inherit the vulnerabilities of both control types.

From the (autonomous) system perspective, there have been multiple reports of (fatal) accidents caused by design flaws/bugs and run-time component failures/faults in autonomous control. Crashes of Boeing 737 MAX [48] and Toyota's "unintended accelera-

tion" [154] are two examples of such cases while the former was caused by sensor failures and the latter is suspected to be the result of a single bit flip. On the other hand, while there is no perfect system that is immune to failures caused by its algorithm design or bugs, human control is also far from being perfect due to the possibility of having unintended errors or even malicious manual control (*e.g.*, Germanwings Flight 9525 [108]).

The replacement of mechanical components with their electronic counterparts also introduces new reliability and security risks because of the adoption of new communication interfaces that can potentially be exploited as attack surfaces. The Jeep Cherokee hack [87] is one of the most widely publicized example in which the vehicle was controlled *remotely* by the hacker's exploitation of the vulnerability in an onboard infotainment system. That is, even if an SA system has a well-designed autonomous control function, it can still make an unsafe decision based on compromised/erroneous input.

1.1.3 Security and Safety Enhancement

As there will never be a perfect system that is free of design flaws and is immune to all the attacks, an SA system must be designed to deal with run-time anomalies to ensure its robustness and safety. Specifically, the enhancement of security and safety of an SA system can also be addressed from three perspectives, each of which is one integrity verification problem that corresponds to one element that can directly influence the operation of SA systems.

- *System Operation.* How can a system confirm the integrity of its operation? That is, how can a system make sure it is acting normally and there is no anomaly or compromised component within the SA system that makes the SA system deviate from its normal behavior.

- *Information from External Entity.* While infrastructure and SA systems may utilize the information from other entities to support their function, the correctness of their function

relies on the integrity of the input data. How can the infrastructure or SA systems ensure that the received information is authentic (*i.e.*, without error and manipulation), and hence can be used in safety-critical applications?

- *Control Decision.* How can the SA system determine whether a control decision is safe to execute under potential data anomalies that feed incorrect information to the SA system? What should the SA system do when it discovers a conflict or disagreement between the control decisions made by a human operator and by its autonomous function since both of them cannot be entirely trusted?

1.2 State-of-the-Art and Design Challenges

We now discuss the prior work and the challenges associated with the three integrity verification problems.

1.2.1 Prior Work of Verification of System Operation

An adversary can exploit the vulnerabilities in communication interfaces to inject messages into the in-system network of the SA system (*i.e.*, a data injection attack) or manipulate the data value to be incorrect (*i.e.*, a data manipulation attack) by directly compromising the components or spoofing the sensor.

Because commercial vehicles are one of the most commonly seen SA systems, Koscher *et al.* [70] conducted a detailed study discussing the potential attack surfaces within a vehicle, and Miller *et al.* [87] demonstrated the scenario where a data injection attack will trigger unexpected vehicle behavior (*e.g.*, turning off the ignition of the car). To detect compromised components within a vehicle system, researchers have also proposed defenses against data injection attacks on Controller Area Network (CAN) bus [22, 92, 93]. They utilize the unique characteristics on the transmission side of CAN messages for their detection. Mutter *et al.* [92, 93] propose the use of transmission pattern (*i.e.*, entropy) of

CAN messages with the same CAN ID to detect if any additional message is injected on the CAN bus. Cho *et al.* [22] utilize the unique voltage profile of ECUs to identify any messages that are not transmitted by their designated ECUs.

However, a pure change of system behavior (*e.g.*, how a vehicle accelerates after the driver applies the gas pedal) and a data manipulation attack from a compromised transmission module will only change the payload value in existing messages, without changing the message transmission pattern on in-system networks. This characteristic makes them more difficult to detect than a usual injection attack. Because a component failure/fault can always be mapped to an equivalent data manipulation attack when observed in data level, most approaches for detecting abnormal component/system behavior can also be utilized for the defense against data manipulation attacks. According to their detection targets, the prior work related to detection of abnormal system behavior can be classified further as either fault detection and isolation (FDI) [8, 18, 30, 47, 55, 89, 97, 99, 143, 156] or general anomaly detection (AD) [2, 5, 22, 23, 45, 50, 79, 90, 92, 93, 101, 102, 124, 144, 146, 153, 157]. FDI approaches focus on the detection of faulty components and their localization. They are widely adopted in both automotive, robotic, and industrial control systems (ICS). On the other hand, AD approaches aim to detect anomalies of any kind in the system, including faulty components and malicious attacks.

While the detection target of AD is a superset of that of FDI, their fundamental difference lies in the assumptions that the anomaly in FDI is the result of faulty/compromised components in a plant (*i.e.*, actuators, processes, and sensors) and the effect of the fault will propagate to other components from its origin and there are known/trustworthy control inputs/setpoints available to the detection system. In contrast, an anomaly in AD can be caused by attacks that only target the data in the cyberspace, making the aforementioned assumptions no longer valid. Also, AD approaches mainly focus on development of efficient observers to detect anomalies while considering the data of interest as a group without further identifying the anomalous source(s). Note that the removal of FDI's as-

sumption of the trustworthy sources will make FDI become AD without the capability of isolating/identifying the source of an anomaly.

While the defense against data-injection attacks has been explored extensively, the approaches for detecting data manipulation or abnormal system behavior have one of the following drawbacks:

- They rely on impractical requirements to function in modern commercial systems, such as cars, because they either require detailed measurements that are not always available to the SA system; or
- They can only achieve moderate performance ($<$80% TPR and $>$15% FPR [144]), thus rendering them infeasible for real-world deployment.

1.2.2 Prior Work of Verification of Received Information

Because SA systems may require support from other entities or infrastructure, there have also been defense schemes proposed for the SA system itself and for the infrastructure to verify data received from other entities. Prior work on integrity verification of received information in cyber-physical systems can be classified into two categories, depending on the methods used to verify the received information.

In the first category, the received system state is inferred from wireless measurements, such as received signal strength (RSS), angle of arrival (AoA), and Doppler shift [1, 118, 126, 147, 151]. Given the relationship between transceiver distance and RSS, Xiao *et al.* [147] and Yao *et al.* [151] proposed infrastructure-assisted and infrastructure-free position verification schemes to address vehicle-to-vehicle sybil attacks, respectively. Similarly, RSS is used by So *et al.* [126] to verify the position information reported in basic safety messages (BSMs). Abdelaziz *et al.* [1] thoroughly analyzed the fundamental limit of AoA estimation and developed an AoA-assisted position verification scheme accordingly based on Wald test statistics. In [118], Schäfer *et al.* proposed the use of Doppler

shift measurements at multiple positions to verify the position and velocity reported by the transmitter.

The second type of scheme attempts to estimate the ground truth of the system state based on multi-modal sensor fusion [13, 24, 127, 129, 152]. Bißmeyer *et al.* [13] proposed a scheme based on particle filters to combine GPS with various sensor readings, such as Radar, to track and verify the motion of neighboring vehicles. Yavvari *et al.* [152] designed a detection system to identify abnormal sensor readings, such as position, speed, and acceleration, reported in BSMs based on the sensor data and vehicular attributes contained in previously received BSMs. Stübing *et al.* [127] proposed a two-stage data verification scheme for position and velocity. To address the incorrect decisions caused by sudden maneuver controls, the authors complement Kalman Filter-based motion prediction with probabilistic maneuver recognition. Choi *et al.* [24] proposed a control invariant (CI) scheme to detect sensor (e.g., GPS and inertial sensors) spoofing attacks. In the CI-based scheme, control invariants and thus a state-space model are first extracted from vehicle's physical properties, the adopted control algorithms, and physical laws, to predict the control output. Then, the expected output is compared with the actual observation to determine if the sensors have been attacked.

As discussed above, most existing schemes either are designed as a fixed system to verify a specific (set of) data or require cooperation (e.g., computation resources or outputs) from other entities to function. While there can be various regulations and requirements for different applications, these deployment characteristics will make the existing schemes not able to function if any of their prerequisites are not met. That is, a practical verification system aiming at supporting SA systems should be flexible and can adapt to different deployment requirements.

1.2.3 Prior Work of Verification of Control Decision

There are also approaches that focus on detection of abnormal control decisions, especially in the (autonomous) vehicle domain. They are usually designed to be deployed in an infrastructure or in a vehicle with communication capability to perform anomaly detection on other entities by capturing the general trend of vehicle behavior in the vicinity and determining if any of the vehicle deviates from this general trend [20, 44, 158]. There are also approaches targeting detection of abnormal control behavior of the ego vehicle (*i.e.*, the SA system itself) by determining if the current control input matches a certain driving pattern (*e.g.*, aggressive or dangerous driving) [4, 21, 60, 71, 74, 76, 80, 121, 123, 125, 160]. However, they can only tell whether current control input poses a potential risk to the vehicle in a *normal operation scenario*, but do not determine whether the vehicle behavior is a reaction to a behavioral context and the aggressive control input is actually safer than maintaining a regular driving pattern.

To the best of our knowledge, no prior work focuses on the detection of human control anomalies that also targets the ego vehicle (*i.e.*, the SA system itself), instead of detecting the misbehavior of other entities, while also considering that there can be system anomalies (*e.g.*, faulty sensor readings and attacks). However, a system must simultaneously consider the detection of human control anomaly for the ego system while considering 1) behavioral and operational context, 2) erroneous system control behavior and faulty components, and 3) malicious attack.

1.2.4 Design Challenges

In general, there are four design challenges in developing practical solutions for the three integrity verification problems in SA systems:

• *(C1) No deterministic normal behavior.* The operation of SA systems is dependent on its operation context. Unlike Industrial Control Systems (ICSes) that usually operates in

controlled environments, SA systems usually operate in diverse environments with different contexts and, as mentioned earlier, those contexts are not necessarily available to the SA system. For example, applying 50% of gas pedal when a vehicle is traveling at 50kph will generate a different longitudinal acceleration depending on whether the vehicle is traveling on a flat road or going uphill. A change of system behavior may be the reason for the change of operation context, a component failure, or even an attack. A system must be able to differentiate them to generate correct results.

• *(C2) Limited data availability.* Not all the information related to the operation state of an SA system can be easily accessed for the following two reasons. First, cost-conscious OEMs often include only those sensors or components that are directly related to the core functions of SA systems to reduce manufacturing cost. For example, while road grade (*i.e.*, inclination) is a major factor that can influence the acceleration of a vehicle, commodity cars usually do not have a built-in inclinometer. Second, due to the real-time communication constraints and the bandwidth limitation in the SA systems, only certain types of data will be transmitted through in-system networks. For example, CAN, the *de facto* standard for in-vehicle communication, has only limited bandwidth, *i.e.* 1Mbps with a high-speed CAN [62] and 5–12Mbps with a CAN-FD [61].

• *(C3) Uncertainty of future deployment.* Unlike mobile devices (*e.g.*, smartphones or tablets) that have unified API supports in OS level (*i.e.*, Android or iOS) for requesting data, SA systems have little to no standards related to what data or hardware should be available in the system. For example, the format and information included in CAN messages in cars are proprietary to, and treated as OEM secrets. The data types available in CAN messages may be different not only between car brands but also even between car models/make of the same brand [26]. That is, designing a system with a fixed assumption of data availability will not be useful to another SA system even if they both belong to the same type of SA system (*e.g.*, cars).

• *(C4) No information can be trusted entirely.* Last but not the least, since there is no unified redundancy design or secure components across the same type of SA system, assuming a fixed trusted source in the system may result in an incorrect system output in an SA system that does not satisfy the requirement. Also, since an adversary is shown to be able to compromise communication interfaces [70, 87, 113], any data may be manipulated and no data can be trusted entirely.

1.3 Thesis Contributions

1.3.1 Thesis Statement

Realizing the security and safety concern in SA systems and the lack of practical systems to address the three essential integrity verification problems, the main objective of this thesis is to develop a suite of mechanisms and systems to detect and further resolve the abnormal behavior and data anomalies that may lead to fatal accidents or critical mission failures while addressing the aforementioned challenges. Specifically, we have the following thesis statement:

Thesis Statement: *Cross-validating sensor data and the operation context can enhance the robustness and safety of semi-autonomous cyber-physical systems.*

1.3.2 Contributions

As shown in Fig. 1.2, there are four inter-related research problems/components in this thesis — **CADD**, **DiVa**, **EDRoad**, and **CADCA** — where each component contributes to solving one of the integrity verification problems in SA systems introduced in Chapter 1.1.3.

From a functional point of view, the proposed systems can be implemented as three (add-on) function blocks embedded in SA systems and their supporting infrastructure. Each function block provides a specific robustness and safety enhancement to the origi-

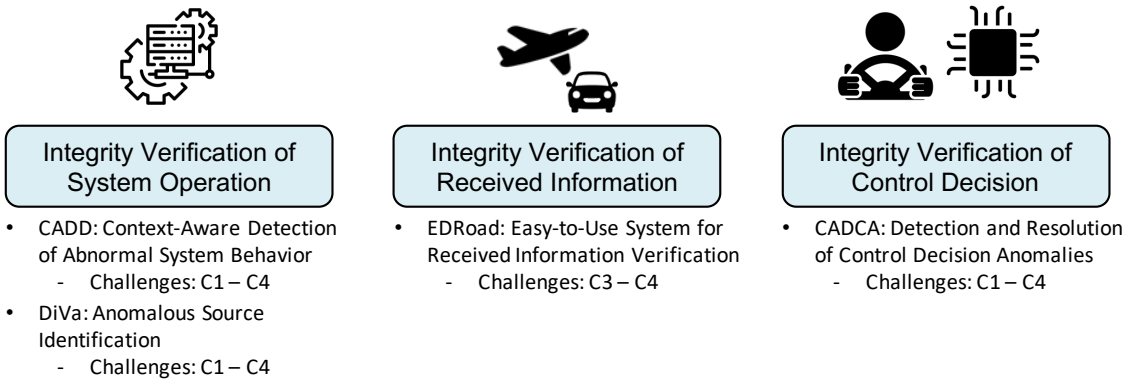


Figure 1.2: Components in this thesis.

nal SA system deployment. Fig. 1.3 shows an overview of how the proposed systems can be integrated in the existing deployment of SA systems, where the white blocks indicate the original components in the SA system and the gray blocks are the new components proposed in this thesis. Specifically, Entity Operation Verifier (Block A), the combination of **CADD** and **DiVa**, is designed for verifying the integrity of system operation; **EDRoad**, the design of Reported Data Verifier (Block B), is designed for verifying the integrity of data received from other entities; and **CADCA**, the design of Control Decision Maker (Block C), is developed to solve the problem of control decision integrity. In each research component, there will be both i) a domain-general design for SA systems and ii) a concrete case study with domain-specific design based on semi-autonomous vehicles.

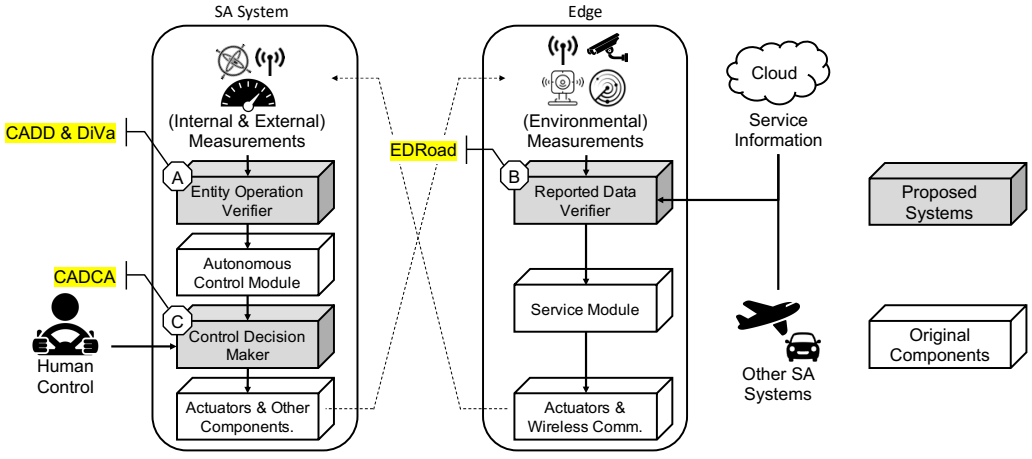


Figure 1.3: Overview of our system design.

We call the combination of all proposed components as **SafeSA**, short for **Safe S**emi-**A**utonomous system. While **SafeSA** is designed to solve the integrity verification problems in SA systems, an anomaly (*i.e.*, the detection target) considered in **SafeSA** is defined as the occurrence of the following (anomalous) conditions.

AC1. System Operation Anomaly:

AC1.1 (Physical Space) A change of the system behavior that does not originate from the adjustment of hardware/settings, the control input from the operator, or the change of execution contexts. For example, if the vehicle changes its acceleration due to an abnormal torque output from the engine, then it will be considered as an anomaly. However, if the low acceleration is caused by driving on an uphill or slippery road, it will not be an anomaly.

AC1.2 (Cyber Space) Any inconsistency between the reported system state and the real one in *data level* while their difference exceeds a predefined error tolerance.

AC2. Received Information Anomaly: Similar to the definition of A1.2, an anomaly in received information is defined as the condition that the received data deviate from their ground-truth values by more than their predefined thresholds.

AC3. Control Decision Anomaly: Any control decision that will lead to an unstable system state, or endangering user safety is considered to be a control decision anomaly. For example, a pilot increases the angle of attack of an airplane too much (*i.e.*, exceeding certain critical value) is considered to be a manual control anomaly because this control can result in a stall (*i.e.*, an unstable system state).

1.3.3 Basic Operation

We now describe the operation of **SafeSA** (Fig. 1.3). Entity Operation Verifier (Block A) is designed to detect and identify anomalies related to the behavior of the ego SA system (Anomalous Condition AC1). It continuously monitors the correlation/causality between

the control inputs and the resulting system state. If any inconsistency is detected, **SafeSA** will further look into whether the change of system behavior is caused by a context change. If yes, Entity Operation Verifier will return to its routine monitoring; otherwise, it will enter diagnostic mode to further identify the origin(s) of the anomaly. Entity Operation Verifier will then output the verified data to the autonomous control system to generate a control decision.

After receiving a control decision, Control Decision Maker (Block C) will determine whether there is certain risks associated with the input control (Anomalous Condition AC3). If there is only one control input and there is no imminent risk associated with its execution, Control Decision Maker will pass the decision to the actuators. If the system receives more than one control input (*e.g.*, there are both autonomous and manual control inputs) with conflicting decisions, it will identify and execute the safer one according to the probable operation contexts.

On the infrastructure side, Reported State Verifier (Block B) continuously detects and corrects the anomalous data received from the SA systems (Anomalous Condition AC2) by cross-validating its sensing results with the received data to prevent the use of incorrect information in services that support SA systems.

Next, we introduce the four components proposed in **SafeSA**.

1.3.4 CADD: Context-Aware Detection of Abnormal System Behavior

The functional goal of Entity Operation Verifier (Block A) is to verify the integrity of system operation. That is, it is designed to detect and locate any anomalies that may indicate a system failure or an attack. Entity Operation Verifier focuses on verifying the correctness of the data measured by the entity itself, including the entity state measurements and the control inputs. There are two parts in the design of Entity Operation Verifier — **CADD** and **DiVa**, where the former is responsible for detecting system control anomalies and the latter is responsible for identifying the source(s) of the anomaly after the anomaly

is detected.

We propose **CADD** to capture abnormal system behavior under various operation context while considering the practical deployment scenarios where some information cannot be directly observed by the SA system. While direct prediction of normal system behavior may be infeasible due to incomplete input information, **CADD** performs anomaly detection based on context estimation, *i.e.* checking whether the operation from different parts of the SA system matches with the same context. By doing so, **CADD** inherently takes context into consideration and do not utilize a fixed trusted source for its detection.

In the case study of SA vehicles, **CADD** determines whether or not the vehicle dynamics match the received control inputs (*i.e.*, gas and brake pedal positions) from the driver or the autonomous driving system under different driving contexts (*i.e.*, road inclination, tire slippage, and total mass). **CADD** utilizes four sets of sensor data to estimate the current context, cross-validates their anomaly detection results, and narrows down the space of search for anomalous groups/components. Our extensive evaluation with $>$87,000 test-cases has shown **CADD** to achieve $>$96% recall and $<$0.5% false positive rate. Furthermore, **CADD** can efficiently pinpoint, or narrow down the anomalous group of data with $>$95% accuracy (*i.e.*, the ratio of correct classifications to the total testing data) when the vehicle's behavior deviates 0.07g from its normal pattern.

1.3.5 DiVa: Anomalous Source Identification

While more and more electronic components are being added to SA systems, their integration becomes complex and introduces new attack surfaces, thus making it harder to identify the source/cause of anomalous system behavior. To mitigate this difficulty, we propose a forensic system, called **DiVa**, that traces down the source(s) of anomalies.

Identification of anomalous source(s) is a crucial functionality in SA systems because this information can be used to determine the level of graveness of the anomaly so that the SA system can respond to it accordingly. **DiVa** constructs hierarchical *detection sets* (*i.e.*,

data groups that describe certain correlation/causality of the system operation) that overlap with each other. By capturing the cyber-physical characteristics in each detection set, **DiVa** is able to cross-validate the data of interest and identify the anomalous one even when no data can be trusted entirely.

In the case study, **DiVa** is designed to identify the source(s) of anomaly related to vehicle maneuvers based on the consistency of input-to-dynamics causality and powertrain operation. Our extensive (with 10,800+ test cases) evaluation shows that **DiVa** is able to pinpoint the exact anomalous data with 0.91 probability under a value-setting attack, and also narrow down the search space to an average of 1.33 sensors with less than 0.1s median detection delay.

1.3.6 EDRoad: Easy-to-Use System for Received Data Verification

While communication between SA entities and infrastructure is essential for enabling advanced applications and services for SA systems, it is important for the infrastructure to verify the integrity of the received data since using incorrect information in services designed to support SA systems can degrade the operation efficiency/security/safety of the SA systems, such as reducing traffic throughput or even causing fatal accidents.

To address this critical problem, Reported State Verifier (Block B) is responsible for detecting and correcting any anomalous data received by the infrastructure. We propose **EDRoad**, a concrete design of Reported State Verifier and an ensemble learning framework for verifying the integrity of *individual* data segments in the received messages and restoring the identified anomalous data, if any. Specifically, **EDRoad** is designed to significantly reduce engineers' development effort, *at design time*, in setting up data verification schemes while meeting the various requirements of different applications.

In the case study of supporting Intelligent Transportation Systems (ITSs) via vehicle-to-infrastructure (V2I) communications, **EDRoad** determines whether or not the vehicle status information (*e.g.*, vehicle location, speed, acceleration, etc.) received by the in-

frastructure is trustworthy and can be used in ITSs. Our extensive evaluation has shown **EDRoad** to achieve >95% recall and <4% fall-out in identifying compromised data in vehicular communication messages and to restore the data in question with a <7% deviation from the ground truth (even when false-positives exist) while providing the salient features that have not been seen/achieved before.

1.3.7 CADCA: Detection and Resolution of Control Decision Anomalies

Since both human and autonomous controls can be erroneous/malicious, we must answer an important question: *which of control decisions should the SA system follow if autonomous and human controls disagree with each other when both of them cannot be trusted entirely and there can be attacks or component failures that feed the system incorrect inputs?*

The answer to this question is not as simple as always choosing the "safer" option because the safety of an action also depends on the context information that needs to be inferred from the perceived data, which can possibly be faulty or compromised as well. While safety features in modern SA systems are usually implemented based on static assignment of control priority, this design may lead to catastrophic accidents when accompanied with erroneous/compromised sensor inputs. **CADCA**, the design of Control Decision Maker (Block C), has the ultimate goal of preventing malicious/erroneous human and system controls to be executed in SA systems. **CADCA** detects both data manipulations and malicious/erroneous control inputs to identify the risks associated with the control inputs. In the case study of semi-autonomous vehicles, our evaluation (>15,700 test-cases) has shown **CADCA** to achieve 98% success rate in preventing the execution of incorrect control decisions caused by component failures and/or malicious attacks in the most common scenarios.

1.4 Outline

The rest of this thesis is organized as follows. Chapter II introduces **CADD**, a system design for context-aware anomaly detection. Chapter III presents **DiVa**, an diagnostic system for identifying the anomalous source(s) after detecting an anomaly. Chapter IV discusses **EDRoad**, an easy-to-use ensemble learning framework specifically designed for verifying SA data on the infrastructure side. Chapter V introduces **CADCA**, a control decision maker that helps SA systems to resolve control conflicts. Finally, this thesis concludes in Chapter VI.

CHAPTER II

CADD: Context-Aware Anomaly Detection

2.1 Introduction

As we are entering the era of semi-autonomous (SA) systems, many components in cyber-physical systems, such as commercial vehicles, are gradually replaced by their electronic counterparts to provide advanced services, such as adaptive cruise control. This decrease of mechanical components makes SA systems increasingly rely on the correctness of their implementation and the messages exchanged within the SA system to perform control-related functions, especially in vehicles with drive-by-wire capabilities. While embracing the convenience brought by the new technologies, there have been multiple incidents of unexpected vulnerabilities caused by the electronic components (*i.e.*, software bugs/glitches and cyber attacks) to the vehicles. The consequences of exploiting such vulnerabilities can range from confusing the driver to jeopardizing the safety of drivers and passengers. One of the most publicized incidents is Toyota's "unintended acceleration", where a single bit flip or a memory/buffer overflow is suspected to lead to fatal accidents [154]. Furthermore, attacks, such as in-vehicle data injection and sensor spoofing, are shown to cause unexpected vehicle behavior [70, 113] or even seize the control of a vehicle [87].

Recognizing these risks of anomalous system behavior, researchers have been exploring ways of characterizing/modeling the normal system behavior that can be used to detect anomalous behaviors, especially for SA vehicles. According to their detection targets,

the proposed solutions can be categorized as the detection of *Message Injection* (MI) or *Data Anomaly* (DA), where the former [3, 22, 92, 93, 145] focuses on the detection of any (malicious) message injection into the in-vehicle network (IVN) and the latter [8, 23, 28, 45, 47, 50, 54, 90, 101, 102, 144, 146, 156] focuses on the detection of anomalies observed in the data level. Specifically, the MI solutions focus on detecting anomalous behavior on the IVN and can achieve high ($> 97\%$) true positive rate (TPR) and low ($< 1\%$) false positive rate (FPR) in exposing the message sent by a malicious/malfunctioning Electronic Control Unit (ECU), but they cannot detect/identify the condition in which only the *system/vehicle* behavior or the data values in messages are changed.

On the other hand, the DA solutions are designed to detect anomalies by capturing inconsistencies between their data of interest. They model the correlation/causality between data and report a misbehavior detection if the correlation between the data does not hold. Therefore, they are capable of detecting the conditions in which the anomaly does not originate from the network level or only the data value in a message is manipulated while other message transmission characteristics remain intact. However, to the best of our knowledge, existing DA solutions, especially the prior work in the vehicle domain, either require detailed in-vehicle measurements that are usually not accessible through the in-vehicle network (*e.g.*, wheel torque [54] and brake torque [23]), or they can only achieve moderate performance when limited/insufficient data availability is taken into account (Chapter 2.2).

There are three major technical challenges in developing an efficient end-to-end DA solution:

C1: *Missing Context Information.* Unlike industrial control systems (ICSs) that usually operate in a controlled environment and are equipped with sensors to capture all major factors of influencing the system state, commercial systems like cars usually operate in diverse environments with different contexts (*e.g.*, road conditions) and, for most of the time, there does not exist any sensor that can detect such context information directly. Furthermore, the SA system depends strongly on its operating environment. For example,

the level of acceleration resulting from pressing a gas pedal at 50% level when the vehicle is traveling steep uphill, will be much less than that of traveling on a flat road. Therefore, it is important for the system to estimate, and account for the "context" for reliable detection of anomalies.

C2: *Limited Data Availability.* Even if the context information can be obtained, the availability of in-system data is often limited, and only coarse-grained or specific data can be accessed. We will use commercial vehicles as a concrete example. First, the standardized on-board diagnostic II (OBD-II) messages based on SAE J1979 only cover the data types related to transmission. Second, CAN, the *de facto* standard for in-vehicle communications, only has limited bandwidth, *i.e.* 1Mbps and 5–12Mbps with a high-speed CAN [62] and CAN-FD [61], respectively. Also, the data types available in CAN messages may be different not only between car brands but also even between car models/make of the same brand [26]. Utilization of detailed measurements that cannot be obtained through standard OBD-II messages or a simple add-on to the vehicle is infeasible for real-world deployment (see Chapter-2.8 for more background information). This fundamental limitation of data availability is the very reason why most, if not all, model-based detection schemes (*e.g.*, system-invariant approaches [25, 104]) have only focused on *robotic* (not commodity) vehicles (Chapter 2.2).

C3: *Training Difficulty.* While detailed internal design of SA systems is available to neither third-party developers, the detection system must learn/estimate the normal behavior of the target SA system based only on partial/incomplete information (C1 & C2) for anomaly detection. Also, the system must be able to work correctly even if it encounters a situation it never saw/experienced before since training for all possible situations is practically impossible. Even if the system is designed as a cloud-based solution to collect data for the training purpose, constructing a unified model that fits all systems with the same model/make is not possible because of customization/adjustment or different permissible error of components in each individual entity.

Realizing the lack of an efficient DA solution, we propose a system, called Context-aware Anomaly Detection in system Dynamics (**CADD**), for detecting anomalies in SA systems under the constraint of limited data accessibility. Unlike traditional approaches that perform detection based on direct comparison between the perceived and the expected system behavior, **CADD** performs its detection based on context estimation. Specifically, **CADD** estimates its operation context from multiple aspects and determines whether the context estimations match with each other. By doing so, **CADD** avoids the possibility that it may make an incorrect normal behavior prediction due to some missing/incorrect context information and output an incorrect detection result.

Based on this domain general design, we also develop a case study based on semi-autonomous vehicles that verifies whether or not the control inputs (*i.e.*, gas and brake pedal positions) from the driver or an autonomous car (*i.e.*, engine torque) [64] match the resulting vehicle dynamics (*i.e.*, acceleration) by determining if there is any vehicular behavior that deviates from its normal operation. Fig. 2.1 depicts the functional overview of **CADD** in this case study. In the case study, **CADD**'s design stresses deployability and practicality based on two design principles.

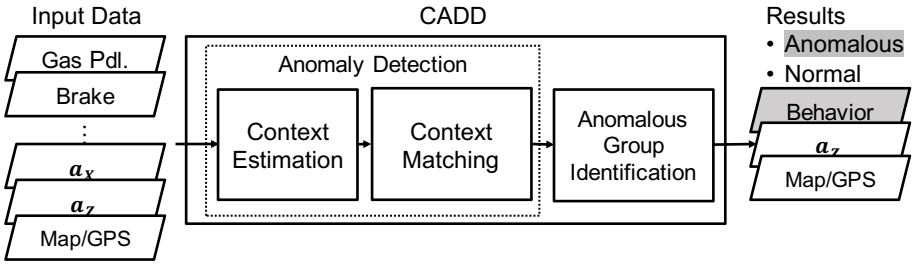


Figure 2.1: Functional overview of **CADD**.

P1: With the goal of requiring no to minimum support from OEMs nor modification to vehicles to facilitate its deployment, **CADD** eliminates the requirement of controller set-point or output, which was commonly required in prior work as input, for the detection of anomalies. That is, neither detailed measurements (*e.g.*, wheel/brake torque) nor vehicle parameters (*e.g.*, gear transition curve) are required for **CADD**'s deployment and the un-

known characteristics of normal behavior considered in **CADD** can all be obtained during its training.

P2: **CADD** considers three fundamental contexts — *road inclination* (RI), *tire slippage* (TS), and *total mass* (TM) of the vehicle including passengers and cargo — that influence a vehicle's longitudinal acceleration behavior regardless of how the vehicle is maneuvered. While these contexts are by no means exhaustive, they are known to be the most influential factors of a vehicle's behavior [53, 65, 69, 81, 82, 91].

To address **C1–C3** and follow **P1–P2**, we propose a detection mechanism based on context estimations, instead of utilizing the common "predict-then-compare" approach (*e.g.*, [28, 104]). **CADD** estimates the context information from four sets of data and then determines if there is an anomaly by checking whether the context estimations are consistent with each other. That way, **CADD** i) eliminates the requirement of knowing the correct context before performing anomaly detection, ii) takes the operation context into account, and iii) does not require a fixed set of trusted sources for its detection (meeting C1 & P2). Also, **CADD** only needs to be trained while the vehicle is traveling on a road with known inclination (*e.g.*, via a map or GPS) but without excessive tire slippage (*i.e.*, a common scenario). The normal operation models constructed by **CADD** are described based on the common mechanical design of modern vehicles and the laws of physics. Therefore, only the vehicle's (not the driver's) behavior will be captured, enabling **CADD** to *automatically* extend the thus-constructed models without requiring any actual training under the exact same condition (meeting C3). Also, other than the optional input (*i.e.*, brake), all required data can be directly obtained from standard OBD-II messages or by an external device (*e.g.*, an OBD-II dongle or smartphone) if some data are not available on IVN/OBD-II messages (meeting C2 & P1).

This chapter makes the following main contributions:

- A novel system, called **CADD**, for anomaly detection with limited data availability based on context estimation (Chapter 2.4).

- A novel system for verifying the relationship between control inputs and vehicle dynamics based on estimated contexts;
  - Efficient models for capturing vehicle normal behavior and context estimation (Chapter 2.5); and
  - Mechanisms for inconsistency detection and anomalous source/group identification (Chapter 2.6).
- Demonstration of **CADD**'s performance in detecting data inconsistencies and anomalous source via extensive evaluation. **CADD** is shown to be able to achieve $>$96% recall and $<$0.5% false-positive rate. **CADD** efficiently identifies the anomalous group of data with $>$95% accuracy (Chapter 2.7).

2.2 Related Work

Since there are already two comprehensive surveys of anomaly detection in *general* control systems [46, 150], we only discuss the approaches directly related to **CADD**.

2.2.1 Data Anomaly (DA) Detection

The DA solutions can be categorized further as *fault detection and isolation* (FDI) [8, 47, 156] or *anomaly detection* (AD) [23, 28, 45, 50, 54, 90, 101, 102, 144, 146]. FDI focuses on the detection and identification/isolation of the faulty source(s) while AD focuses only on the detection of abnormal behavior caused by both component failures and malicious attacks.

The most common framework used in FDI describes the system of interest using techniques like Bond Graph [156], and formulates the causal relationships between components as a system with a set of equations. Each equation in the detection system is an observer designed to capture specific aspects of failures. While treating the failures as unknown variables, the source of failure(s) can be identified/isolated efficiently by solving the system equations [8, 47, 156]. FDI focuses on modeling the interaction between components

while requiring detailed component design or architecture to function.

AD exploits the correlation between vehicle data to construct a normal behavior model for anomaly detection. The authors of [50, 90, 101, 102] focus on the detection of the engine while Xi *et al.* [146] and Cho *et al.* [23] focus on the transmission and brake system, respectively. While these studies focus on a single functional group, Ganesan *et al.* [45] proposed an anomaly detection system that covers multiple functional groups. They considered pair-wise data correlation and used clustering to determine the context (*i.e.*, traffic pattern) for the detection of abnormal vehicle behavior. Similarly, Guo *et al.*[54] proposed a detection system, *EVAD*, which is reported to achieve 98.8% TPR and 1% FPR based on pair-wise correlation of detailed in-vehicle measurements. However, some detection pairs in [54] also utilize data that are not commonly available on IVNs, such as ⟨wheel torque, acceleration⟩ and ⟨brake torque, brake pedal⟩. [144] proposed using neural networks to detect anomalies in vehicle speed and engine rpm/torque. However, it is reported to achieve only moderate performance with $<$80% TPR and $>$15% FPR in detecting anomalies. Dash *et al.* [28] proposed a detection system, called *PID-Piper*, based on a long short-term memory (LSTM) learning architecture for control behavior monitoring.

Since (i) PID-Piper is one of the latest control invariant (CI) approaches based on machine learning and (ii) EVAD [54], a correlation-based approach, covers data types most similar to those in **CADD** and, to the best of our knowledge, provides the best performance among all prior work, we will use them for a baseline comparison in Chapter 2.7.

2.2.2 Context Estimation (CE)

We now discuss the related work of Context Estimation in vehicle domain. Since road inclination (RI) and tire slippage (TS) caused by insufficient road friction (RF) are very important for control systems to enhance system stability and reduce fuel consumption, several approaches have been proposed for their estimation. Mangan *et al.* [82] proposed a method utilizing a vehicle's physical/kinematic model to estimate RI based on the vehi-

cle's speed, longitudinal acceleration, brake pressure, and engine torque by computing the difference between the expected acceleration on a flat road and the actual measured acceleration. Mahyuddin *et al.* [81] proposed a method that utilizes vehicle speed and driving torque, and adopts filtering and adaptive observer techniques for the estimation. Jauch *et al.* [65] proposed an IMU-based method for RI estimation. Specifically, RI (θ) is estimated based on i) the difference between measured longitudinal acceleration and the derivative of the vehicle's speed relative to the ground: $\theta = \arcsin[(a_X - dv/dt)/g]$; and ii) altitude difference Δh (obtained from GPS) divided by the traveled distance $\Delta \ell$: $\theta = \arcsin[\Delta h/\Delta \ell]$. For pure RF estimation, Muller *et al.* [91] proposed a slip-based approach to estimate the maximum friction when the brake is pressed. Similarly to [65], the authors of [69] and [53] utilized measurements of vehicle dynamics along with filtering techniques to estimate RI and RF/TS.

2.2.3 Comparison of CADD's Case Study with Prior Work

1) System Scope: To the best of our knowledge, there is no prior work on data anomaly (DA) detection that has the same system scope as **CADD** — a model-based end-to-end (*i.e.*, control-to-dynamics) detection with driving context (*i.e.*, road inclination, tire slippage, and mass change) considered.

While many prior approaches utilize physical consistency between (vehicle) dynamics data for anomaly detection (*e.g.*, vehicle speed should be consistent with vehicle acceleration, etc.), they are only applicable to scenarios in which the attack/anomaly will cause inconsistencies between different sensor data. However, since the detection target of **CADD** is the vehicle's control-to-dynamics behavior (Chapter 2.3.1), relying only on physical consistency modeling cannot cover this detection target. For example, if a component failure causes the vehicle to generate less engine torque than it normally does, the vehicle speed and acceleration will still match each other. On the other hand, pure machine learning approaches cannot efficiently adapt to different driving contexts that are not included in

their training data (to be shown in Chapter 2.7). Therefore, **CADD** models the vehicle's (acceleration) behavior based on the common vehicle architecture and utilizes supervised machine learning to capture the detailed individual behavior that may vary from vehicle to vehicle. Also, instead of directly comparing the predicted and the received vehicle behavior, **CADD** performs its detection by determining whether all data point to the same driving context, avoiding inaccurate behavior prediction with insufficient/compromised data.

2) Context Awareness: There is also no easy way to combine DA and CE to form a context-aware anomaly detection because the latter usually assumes the detailed vehicle parameters and operation mechanisms known to the estimation system (*e.g.*, gear and final drive ratio in [82]) while the former does not have such information. Since prior DA approaches do not consider CE in the first place, they do not have any the training mechanism to obtain those parameters either. Besides the availability of necessary information, a detection system must also consider all three major contexts *simultaneously*. However, incorporating all three contexts in a detection system is not as easy as including different CE approaches in the system. For example, [69] can be used to perform road inclination estimation and detect tire slippage while [81] focuses on the estimation of vehicle mass. However, since [81] is not designed to operate when a tire slippage occurs, a combination of the two approaches cannot be utilized to detect anomalies when there is a tire slippage. Thus, prior CE approaches cannot be applied directly to **CADD** for context estimation. As a result, instead of trying to utilize the models in prior CE approaches, we develop behavior models to account for the effect of driving context to address the aforementioned challenges. The proposed models in **CADD** can not only capture the vehicle's normal behavior under different contexts but also be used to perform context estimation. That way, all the necessary parameters can be obtained from **CADD**'s training phase, facilitating its deployment.

3) Data Requirement: **CADD** is designed based on limited data accessibility, and requires minimum deployment effort and no to minimum support from the first party. Due

to the nature of their design, prior DA approaches, including control/system invariant and residual design [7, 25, 39, 137], focus on mechanisms that are inherently tied to their required data, and hence there is no easy way to change them without re-designing the system. While the data commonly available on IVNs are control inputs and dynamic measurements, the lack of final control output, such as wheel torque, will render the aforementioned prior work ineffective or dysfunctional.

2.3 Threat Model and Detection Scope

2.3.1 Detection Scope

Because, we will use a case study of SA vehicles to illustrate **CADD**'s design, we first introduce terminologies to be used throughout the chapter. We define the end-to-end (*i.e.*, control-to-dynamics) acceleration behavior of a vehicle as the *vehicle's behavior* (VB). Let *Data Of Interest* (DOI) be the data or measurements that **CADD** uses in its anomaly detection, including:

- *Major Detection Target (i.e., VB data)*:
  - *Control Input*: gas pedal or throttle position (g_a), brake pedal position or master cylinder pressure (b_r)[1], gear level (g_r), and engine torque (T_q);
  - *Dynamics Measurements*: longitudinal acceleration (a_X), and speed (v);
- *Assistance Data*: vertical acceleration (a_Z) and a sensor for road grade (θ_d) estimation, *e.g.*, GPS (standalone or with maps) or inclinometer.

Specifically, g_a, T_q, g_r and v can be retrieved directly from standard OBD-II messages [115] and the rest are commonly available on IVN in drive-by-wire vehicles [26]; otherwise, a_X, a_Z, and GPS can also be obtained from an external device like a smartphone. As mentioned earlier, unlike prior work, **CADD** does *not* assume access to the final system output (*e.g.*, wheel/brake torque) since it is not commonly available.

[1] b_r is treated as an optional target data since it is not included in the standard OBD-II message.

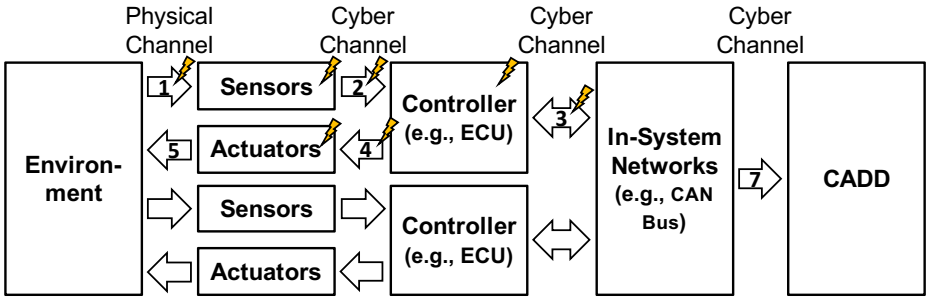


Figure 2.2: Information flow in **CADD**, where the lightning icons are the anomaly (attack) interfaces covered by **CADD**.

CADD is designed to detect VB anomalies in the *data level* (Fig. 2.2). An *anomaly* (*i.e.*, the detection target) considered in **CADD** is defined as the occurrence of:

A1. A change of the vehicle's response to control input that does not come from the change of driving contexts (*i.e.*, RI, TS and TM); or

A2. (Cyber Space) An inconsistency between the VB pattern and the driving context in the *data level*.

For example, if the vehicle changes its acceleration due to an abnormal torque output from the engine, then it will be considered as an anomaly. However, if the low acceleration is caused by driving on an uphill or slippery road, it would not be an anomaly. For A1, longitudinal acceleration (a_X) deviating by more than xg from its normal value is defined as an anomaly, where x is a design parameter and g is the gravitational acceleration. Similarly, input information of road inclination (θ_d) deviating by more than $y°$ from its ground truth is defined as an anomaly in A2, where y is also a design parameter related to x. To see the effects of these design parameters, we will in Chapter 2.7 evaluate **CADD**'s performance while changing x and y. Note that **CADD**'s case study focuses on a vehicle's control-to-dynamics behavior, but *not* on how the autonomous system reacts to different traffic conditions or faults/attacks that do not change VB.

2.3.2 Threat Model

In addition to component failures that fundamentally change the vehicle's response to control inputs (*i.e.*, A1), we assume the adversary has the goal of influencing the state estimation/prediction of the vehicle to make it deviate from its set course or waypoints and launches *remote* attacks by either i) spoofing sensors (Arrow 1 in Fig. 2.2) or ii) manipulating a set of chosen DOIs on in-vehicle networks to any value s/he wants (Arrow 3). **CADD** is designed to detect anomalies when at least one status measurement (*i.e.*, dynamic measurements and assistance data defined in Chapter 2.3.1) is correct without the requirement of knowing which one. Note that the above setting is a fundamental requirement of all data-driven approaches (*i.e.*, not just **CADD**). That is, there must be at least one degree of freedom (DoF) in the target system that the attacker cannot fully control. Finally, **CADD** is designed to identify the anomalous source when only vehicle's response to control input or one of the assistance data is compromised (*e.g.*, a component fault or a naïve attack).

This threat model meets the real-world condition of commercial vehicles (or SA systems in general) for the following reasons. First, while spoofing GPS is shown to be plausible from a distant location, spoofing an accelerometer to generate a specific waveform (*i.e.*, not random values) requires a learning process for phase tuning based on the feedback from the accelerometer itself [133]. That is, spoofing an accelerometer is proven difficult without direct physical access to the vehicle's internal component. Second, while prior work [87] only showcases the possibility of data injection/manipulation through *a single ECU* attached to the infotainment system, using a single ECU to mimic data transmission of multiple ECUs can be prevented with MI approaches [3, 22, 145]. Third, since most ECUs have limited communication capability, attackers must have physical access to those internal ECUs to manipulate their core operation. Nevertheless, we still assume a stronger attacker (than existing attack examples) to have the capability to manipulate all data but one status measurement.

2.4 System Overview

2.4.1 Domain-General Design

We propose a detection mechanism based on context estimations, instead of the direct predict-then-compare approach. There are two phases in **CADD**'s basic operation: *training* and *detection* (Fig. 2.3). During the training phase, **CADD** captures SA system's basic operation and builds multiple models for context estimations from different aspects. In the detection phase, **CADD** first detects whether there is an anomaly and further identifies whether the anomaly is caused by an abnormal system behavior or incorrect context information.

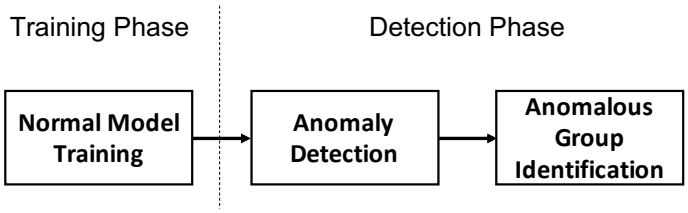


Figure 2.3: Basic operation of **CADD**.

Specifically, every model constructed in the training phase is presented by specific groups of data, called *context estimation groups* (CEGs), each of which estimates a certain operation context of the SA system. If the estimated contexts are not consistent with each other, **CADD** will report an anomaly detection. This way, **CADD** takes the context information into account and does not require any trusted input for context estimation to perform its detection as in traditional predict-then-compare approaches. Fig. 2.4 shows the block diagram of **CADD**'s detection procedure. There are two major function blocks — Context Estimator and Context Matcher. The former is responsible for capturing common contexts while the latter is responsible for detecting special cases and reporting the occurrence of an anomaly.

In Context Estimator, **CADD** performs context estimations based on the models ($\mathcal{M}_{CEG}$'s) that capture the correlation between CEGs and the target context. For example, the road

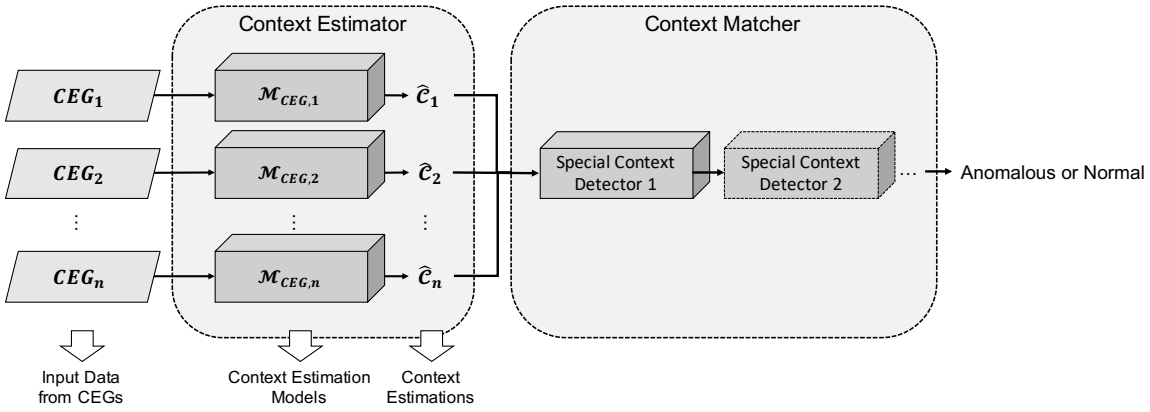


Figure 2.4: **CADD**'s domain-general design.

inclination θ (*i.e.*, the target context) can be estimated based on i) the difference between the expected longitudinal acceleration when a vehicle is travelling on a flat road ($\hat{a}_X$) and the measured longitudinal acceleration (a_X): $\hat{\theta} = \arcsin((\hat{a}_X - a_X)/g)$ or ii) the vertical acceleration (a_Y) compared to the gravitational acceleration (g): $\hat{\theta} = \arccos(a_Y/g)$. After context estimations are made, Context Matcher will further compare the estimations ($\hat{\mathbb{C}}$'s) to see whether they match each other. If not, Special Context Detector will proceed to check whether the pattern of the mismatch between the estimations matches with a special context. If no such condition can be identified, Special Context Detector will conclude that there is an anomaly and activate **CADD**'s anomalous group identification process.

2.4.2 Domain-Specific Design — a Case Study of Commercial Vehicles

We now introduce how to utilize **CADD** to build a system for detection of vehicle maneuver anomalies. During the training phase, **CADD** captures a vehicle's basic operation of regular driving. No specific constraint is imposed on how the driver/system should maneuver the vehicle as long as there is no tire slippage during the training phase which is the usual/regular driving scenario. Furthermore, according to our experimental results, a 5km drive suffices for capturing the vehicle's normal operation. Ideally, this training needs only once for each car unless some physical modification or maintenance is made to the

car. The user can also choose to perform training periodically to account for component aging/wear-out. However, it is the user's choice whether to do so since a positive detection reported by **CADD** can also be the indication that the vehicle requires maintenance.

The normal vehicle's behavior can be described by two models, $\mathcal{M}_{gb}$ and $\mathcal{M}_{tg}$. The former captures the relationship between longitudinal acceleration and the drivers' control input (Chapter 2.5.2) while the latter captures the relationship between longitudinal acceleration and the engine torque, which is the common output for speed control [64] (Chapter 2.5.3).

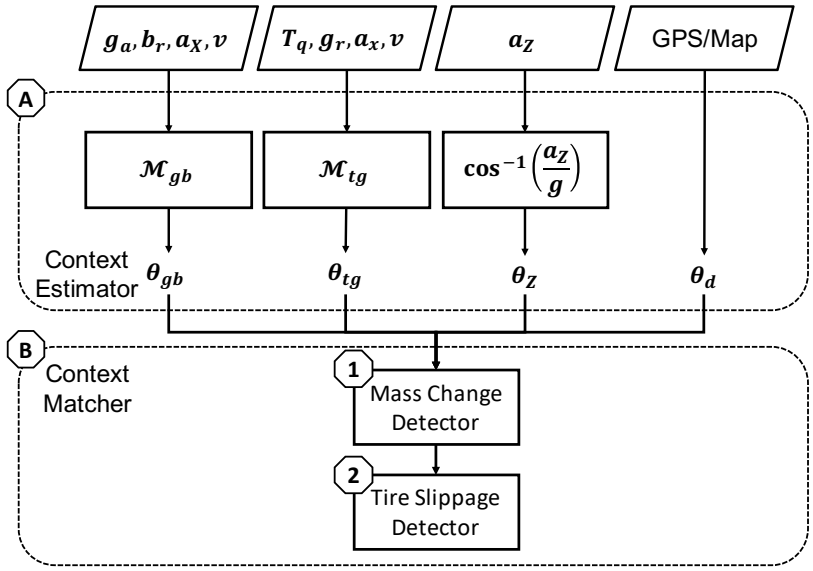


Figure 2.5: System structure of **CADD**'s detection phase of the case study.

Fig. 2.5 shows the system structure for the detection phase. **CADD** performs the anomaly detection by estimating the contexts (*i.e.*, RI, TS, and TM) and comparing them to check if they are consistent with each other. Specifically, **CADD** first assumes that all vehicle's behavioral changes are caused by the road inclination (RI). So, **CADD** estimates RI from four perspectives independently: (i) the difference between measured (longitudinal) acceleration (a_X) and expected acceleration based on gas/brake pedal position input ($\hat{a}_{X,gb}$); (ii) the difference between a_X and expected acceleration based on engine torque and gear level ($\hat{a}_{X,tg}$); (iii) RI information provided by GPS/map; and (iv) the difference

between measured vertical acceleration (a_Z) and gravity (g). If these four estimations of RI match each other, **CADD** will conclude that the vehicle behaves normally. Otherwise, **CADD** will check further if the change of the vehicle's behavior is caused by the other two contexts (*i.e.*, TS and TM) by cross-validating the RI estimations and the control input data.

If the anomalous behavior is determined not caused by any of the contextual changes, **CADD** will report the detection of an anomaly and send the potential anomalous data/component group to the user or the autonomous driving system. The choice of data utilized in **CADD** is based on their functional roles in the speed control and their accessibility. We chose gas and brake pedal positions as they are the direct inputs from the drivers. Also, engine torque and gear level can be used to derive the wheel torque, which is a common output of the speed control system [64].

2.5 Normal Behavior Model

2.5.1 Fundamentals of Vehicle Acceleration

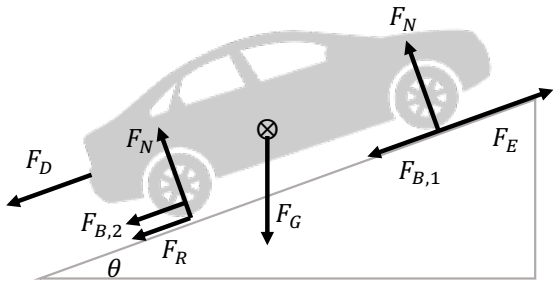


Figure 2.6: Forces related to vehicle acceleration.

In general, there are six types of forces that can directly influence a vehicle's acceleration (Fig. 2.6) [82]:

- Drive force (F_E): force generated from engine torque (T_q);
- Brake force (F_B): force generated from braking;
- Aerodynamic drag (F_D): force caused by air resistance;

- Rolling drag (F_R): force required for tires to roll passively;
- Normal force (F_N): supportive force perpendicular to the contact surface; and
- Gravitational force (F_G): force caused by gravity.

Therefore, the vehicle acceleration **a** can be described by:

$$m\mathbf{a} = \mathbf{F_E} + \mathbf{F_B} + \mathbf{F_R} + \mathbf{F_D} + \mathbf{F_N} + \mathbf{F_G} + \mathbf{F_O}, \tag{2.1}$$

where m is the vehicle's total mass, F_O is the aggregated effect from other minor factors, and the terms in bold fonts represent vectors. If we look at the vehicle's longitudinal direction (presented by subscript X) and plug it in the detailed expression of each force based on the available DOI of **CADD**, then Eq. (2.1) can be rewritten as Eq. (2.3) when $|\mathbf{F_E} + \mathbf{F_B}| \leq \mu F_N$:

$$\begin{aligned} m_{(\Sigma)}a_X &= F_E - F_B - F_R - F_D - F_{G,X} + F_{O,X} && (2.2)\\ &= \frac{T_q i_g i_f}{R_{(\Sigma)}} S_{E(\Sigma)} - b_r k_{b(\Sigma)} S_{B(\Sigma)} - f_{(\Sigma)} m_{(\Sigma)} g\cos\theta \\ &\quad - 0.5\ \sigma_{(\Sigma)} c_{(\Sigma)} A_{(\Sigma)} v^2 - m_{(\Sigma)} g\sin\theta + \psi_{(\Sigma)}, && (2.3) \end{aligned}$$

where μ is the coefficient of friction between the tires and the road surface, i_g is the gear ratio, i_f is the final drive ratio, and R is the tire radius. k_b is the brake force coefficient, and S_E (S_B) is the adjustment on F_E (F_B) due to vehicle steering. f is the coefficient of the vehicle's rolling resistance. σ is the air density, c is the vehicle's drag coefficient, and A is the vehicle's cross-section area. Finally, ψ is the aggregated effect of other minor factors, and the terms with subscript (Σ) indicate that their values can be influenced by contexts/operation that are not available to **CADD**.

While performing anomaly detection with complete cyber-physical information is rather easy/trivial, it is a major challenge that how can **CADD** perform anomaly detection when only limited data can be accessed. Our contribution lies in developing novel mechanisms

and models that describes vehicle's operation behavior for context estimation and end-to-end anomaly detection with such a constraint. The unobservable adjustments caused by (Σ) will be treated as model noise (*i.e.*, value deviation between the constructed model and the ground truth model caused by not having enough data during training or execution) during deployment. However, they can be incorporated if they become standardized data. See Chapter-2.8 for more discussion. To facilitate a more concise presentation, we omit (Σ) in the following sections.

2.5.2 $\mathcal{M}_{gb}\{g_a, b_r, v, a_X\}$

1) Model Formulation: $\mathcal{M}_{gb}$ captures the relation between a_X and drivers' control inputs (*i.e.*, g_a and b_r). We first combine F_D, F_R, and $F_{G,X}$ together:

$$F_R + F_D + F_{G,X} = (fmg\cos\theta + \frac{\sigma c A v^2}{2}) + mg\sin\theta \tag{2.4}$$

$$\approx \underbrace{fmg + 0.5\sigma c A v^2}_{\mathcal{T}_{R,D}(v)} + \underbrace{mg\sin\theta}_{\mathcal{T}_G(\theta).} \tag{2.5}$$

We can make the approximation in Eq. (2.5) because $\tan\theta \leq 0.07$ according to the US government's guideline for road construction [6]. We then use the function form to present $F_E = \mathcal{T}_T(T_q, g_r)$ and $F_B = \mathcal{T}_B(b_r, v)$. Since the engine torque (T_q) is controlled by the throttle while gear ratio is determined by the gear level (g_r) which is correlated with the vehicle speed (v) and throttle position (g_a), F_E can be described by a function of g_a and v. Finally, we can obtain the model formulation of $\mathcal{M}_{gb}$ by plugging Eq. (2.5) into Eq. (2.3):

$$a_X = \frac{1}{m}[\mathcal{T}_T(T_q, g_r) - \mathcal{T}_B(b_r, v) - \mathcal{T}_{R,D}(v) - \mathcal{T}_G(\theta)] \tag{2.6}$$

$$= \underbrace{\mathcal{H}_T(T_q, g_r)}_{\mathcal{H}_E(g_a, v)} + \mathcal{H}_B(b_r, v) + \mathcal{H}_{R,D}(v) + \mathcal{H}_G(\theta) \tag{2.7}$$

where $\mathcal{H}_i$ is the aggregated effect of $\mathcal{T}_i$ on a_X.

2) Training: To simplify the model training process, we divide the $\mathcal{M}_{gb}$ model into two

submodels — $\mathcal{M}_u$ and $\mathcal{M}_d$, where the former focuses on modeling the vehicle's behavior when only gas pedal is applied and the latter focuses on any other situation. Since $\mathcal{H}_G$ is not dependent on any vehicle state, we can compute its effect on a_X by training on any road with a known slope. Also, since modern automatic transmission is usually controlled based on vehicle speed (v) and throttle position (g_a) [146], the aggregated effect of $\mathcal{H}_T(T_q, g_r) + \mathcal{H}_{R,D}(v) = \mathcal{H}_E(g_a, v) + \mathcal{H}_{R,D}(v) = \Gamma(g_a, v)$ can be obtained by only considering the training data when the brake is not pressed. So, we can now express the simplified model $\mathcal{M}_u$ as:

$$a_X = \Gamma(g_a, v) + \mathcal{H}_G(\theta), \text{ if } b_r = 0. \tag{2.8}$$

Training $\mathcal{M}_u$ is to identify Γ given the training data $\{g_a[t], v[t], a_X[t]\}$. Since the acceleration characteristics vary with vehicles, we do not use a fixed model form to approximate the function Γ. Instead, we model the VB based on the vehicle's speed. That is, we assign a function Γ_k to describe the correlation between g_a and a_X in each speed interval $v \in [v_k, v_{k+1})$, where $v_k = \delta_v \times (k-1)$ and δ_v is the size of each group (Fig. 2.7a). This way, we can model Γ without committing to a specific model form and achieve the flexibility of this approach. We set $\delta_v = 5$km/h in our implementation.

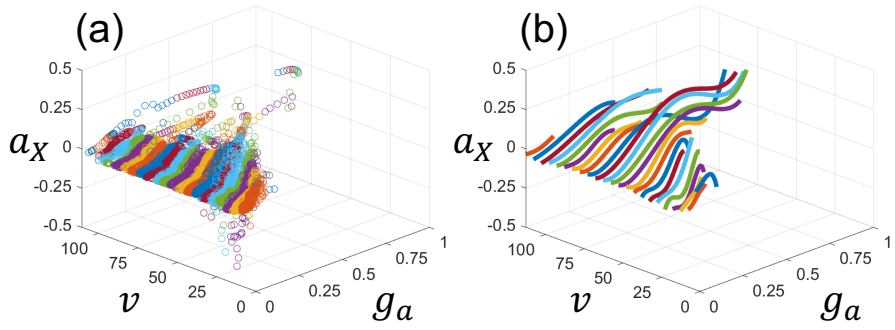


Figure 2.7: (a) The acceleration characteristics partitioned by v. (b) $\mathcal{M}_u$'s final model example.

Since any function can be approximated by its Taylor expansion form, we use a poly-

nomial approximation:

$$\Gamma_k(g_a[t]) = c_{k,0} + c_{k,1} g_a[t] + c_{k,2} g_a^2[t] + \cdots, \tag{2.9}$$

where coefficients $[c_{k,0}, c_{k,1}, c_{k,2}, \cdots]^T = \mathbf{c_k}$ are determined by minimizing the loss function:

$$L_k = \Sigma_{t_0}^{t_n} (a_X[t] - \mathbf{c_k}^T \mathbf{G_a}[\mathbf{t}])^2 + \lambda |\mathbf{c_k}|^2, \tag{2.10}$$

where $v[t] \in [v_k, v_{k+1})$ and $\mathbf{G_a}[t] = [1, g_a[t], g_a^2[t], \cdots]^T$. Fig. 2.7b shows the final model form, where the lines are Γ_k in the speed interval k.

We use another model $\mathcal{M}_d$ to capture VB when the brake is applied:

$$a_X = \mathcal{H}_{R,D}(v) + \mathcal{H}_E(g_a, v) + \mathcal{H}_B(b_r, v) + \mathcal{H}_G(\theta) \tag{2.11}$$

$$= \Gamma(g_a, v) + \mathcal{H}_B(b_r, v) + \mathcal{H}_G(\theta) \tag{2.12}$$

Since $\mathcal{H}_G(\theta) = -g \sin\theta$ can be compensated by performing the training process on a road with known inclination, the main idea is to determine the parameter values under different speed conditions by approximating Eq. (2.12) as we did to obtain $\mathcal{M}_u$:

$$a'_X = b_{X,o,0}(v) + \sum_{i=1}^{N_g} b_{X,g,i}(v) g_a^i + \sum_{j=1}^{N_b} b_{X,b,j}(v) b_r^j \tag{2.13}$$

where $a'_X = a_X - \mathcal{H}_G(\theta)$, and N_g (N_b) are the orders of Taylor expansion forms of Γ ($\mathcal{H}_B$). Theoretically, we can perform a similar procedure of $\mathcal{M}_u$ to obtain the parameters. However, because the amount of time the brake is pressed is usually only a small portion of time during driving, the amount of total training data needed to construct a usable model will be huge according to our preliminary experimentation. Therefore, we treat the parameters (b_X's) as functions of v to make up for the missing scenarios in the training data instead of directly training them for every v interval as in $\mathcal{M}_u$.

CADD partitions the data into small segments by a fixed time window size (T_w). For each segment, the next step is to perform the ridge regression [12] to estimate $b_{X,o,0}$, $\{b_{X,g,n}\}$, and $\{b_{X,b,n}\}$ in this segment:

$$\hat{b}_k = \text{argmin}_{b_k}(a'_{X,k} - \mathbf{b_k}^T \boldsymbol{\omega}_\mathbf{k})^2 + \lambda |\boldsymbol{\omega}_\mathbf{k}|^2, \tag{2.14}$$

where k is the index of the segment,

$\mathbf{b_k} = [b_{(X,o,0,k)}, b_{(X,g,1,k)}, \ldots, b_{(X,g,N_g,k)}, b_{(X,b,1,k)}, \ldots, b_{(X,b,N_b,k)}]^T$,

$\boldsymbol{\omega}_\mathbf{k} = [1, g_a(t_{k,1}), \ldots, g_a(t_{k,N_T}), b_r(t_{k,1}), \ldots, b_r(t_{k,N_T})]^T$,

N_T is the total number of data items in the segment, $a'_{X,k} = a_{X,k} - \mathcal{H}_G(\theta_k)$, and λ is the regularization term. Finally, **CADD** performs regression again to identify the mapping from speed to the values of parameters.

Note that our implementation uses $\mathcal{M}_u$ to model the VB when only the gas pedal is pressed, and uses $\mathcal{M}_d$ for other situations. This design is based on the fact that the formulation of $\mathcal{M}_u$ is capable of capturing more detailed acceleration behavior resulting from the application of gas pedal than $\mathcal{M}_d$.

2.5.3 $\mathcal{M}_{tg}\{T_q, g_r, v, a_X\}$

1) Model Formulation: $\mathcal{M}_{tg}$ captures the relation between engine output and a_X. When the brake is not pressed, the formulation can be directly obtained from Eq. (2.3) as:

$$a_X = \underbrace{\mathcal{H}_T(T_q, g_r) + \mathcal{H}_{D,R}(v)}_{\mathcal{H}_{T,g_r}(T_q,v)} + \mathcal{H}_G(\theta) \tag{2.15}$$

where $\mathcal{H}_{T,g_r}$ is the aggregated acceleration component contributed by engine torque at gear level g_r.

2) Training: Training $\mathcal{M}_{tg}$ is equivalent to identifying $\mathcal{H}_{T,g_r}$ for each gear level. We

can further express $\mathcal{H}_{T,g_r}$ as:

$$\mathcal{H}_{T,g_r}(T_q, v) = h_{g_r,0} + h_{g_r,1}T_q + h_{g_r,2}v + h_{g_r,3}v^2. \tag{2.16}$$

Similarly to the training of $\mathcal{M}_{gb}$, we can utilize the ridge regression to obtain $\mathbf{h_{g_r}} = [h_{g_r,0}, h_{g_r,1}, h_{g_r,2}, h_{g_r,3}]^T$ by minimizing the following loss function:

$$L_j = \Sigma_{g_r[t]=j}(a_X[t] - \mathbf{h_j}^T\mathbf{Y}[t])^2 + \lambda|\mathbf{h_j}|^2, \tag{2.17}$$

where j is the target gear level and $\mathbf{Y} = [1, T_q, v, v^2]^T$.

2.5.4 RI Estimation based on $\mathcal{M}_{gb}$ and $\mathcal{M}_{tg}$

Using $\mathcal{M}_{gb}$ and $\mathcal{M}_{tg}$ for RI estimation is simple. Since we can separate the acceleration contribution of RI (*i.e.*, $\mathcal{H}_G(\theta)$) from other DOI in the model formulation, all **CADD** has to do is to compute the expected acceleration ($\hat{a}_X$) based on DOI while assuming $\theta = 0$, and take an arc sine of the difference between $\hat{a}_X$ and the actual a_X divided by gravity:

$$\hat{\theta} = \sin^{-1}[(\hat{a}_X - a_X)/g]. \tag{2.18}$$

This estimation is valid only when TM remains the same as the training phase, and there is no tire slippage. If TM changes, say from m to m', the expected acceleration will need to be calibrated by a factor of m'/m to obtain the correct slope estimation (Eq. (2.3)). If there is a tire slippage, the total force can be generated from F_E and F_B will be capped at μF_N. Therefore, pressing more gas or brake pedal will not generate higher acceleration and this will lead to a larger/smaller RI estimation magnitude than the actual value, depending on which control input is applied. **CADD** utilizes this characteristic in its anomaly detection to identify whether a TM change or tire slippage occurs by comparing the RI estimations from VB models with that from vertical acceleration (*i.e.*, $\theta_Z = \cos^{-1}(a_Z/g)$) and RI directly

obtained from GPS/map (θ_d). Our preliminary evaluation shows that $\mathcal{M}_{gb}$ and $\mathcal{M}_{tg}$ are able to capture the vehicle behavior accurately and produce RI estimation with merely 0.1° median error.

2.6 Context-Aware Anomaly Detection

2.6.1 Detection Procedure

CADD performs anomaly detection based on the context estimations. As shown in Fig. 2.5, **CADD** first estimates RI utilizing the VB models while assuming the vehicle is traveling without tire slippage and the mass in the vehicle remains the same as the training phase. If the estimated RIs do not match each other, **CADD** further looks into the data to see if the inconsistency is the result of other contexts. Specifically, **CADD** first estimates the RI from four perspectives (Chapter 2.5.4): i) θ_{gb} from $\mathcal{M}_{gb}$, ii) θ_{tg} from $\mathcal{M}_{tg}$, iii) θ_Z from a_Z, and iv) θ_d from GPS/map (Block A in Fig. 2.5). If the estimations match each other (*i.e.*, maximum and minimum differences of the estimations are smaller than a threshold η_{RI}), **CADD** will report the pass of verification and keep monitoring the DOI. Otherwise, **CADD** looks further into whether it is the change of TM that leads to the inconsistencies of slope estimations (Block B in Fig. 2.5). Note that η_{RI} is a design parameter that can be set by developers according to their preference. We will discuss more on η_{RI} in Chapter 2.7.

As discussed in Chapter 2.5.4, θ_{gb} and θ_{tg} will be different from θ_Z and θ_d if TM changes. However, θ_{gb} and θ_{tg} will match each other since they are both estimated from the difference between $\hat{a}_X$ and a_X. Therefore, **CADD** utilizes this characteristic to check whether $\{\theta_{gb}, \theta_{tg}\}$ and $\{\theta_Z, \theta_d\}$ form two separate groups to determine if the mismatch of RI estimation is caused by TM change. Since a mass change other than fuel consumption[2] is not likely to occur during a trip (*i.e.*, from the vehicle starts to move to the vehicle stops moving), **CADD** filters out the false positives caused by the mass change due to passengers

[2]Negligible during a short trip. See Chapter 2.8 for more on this.

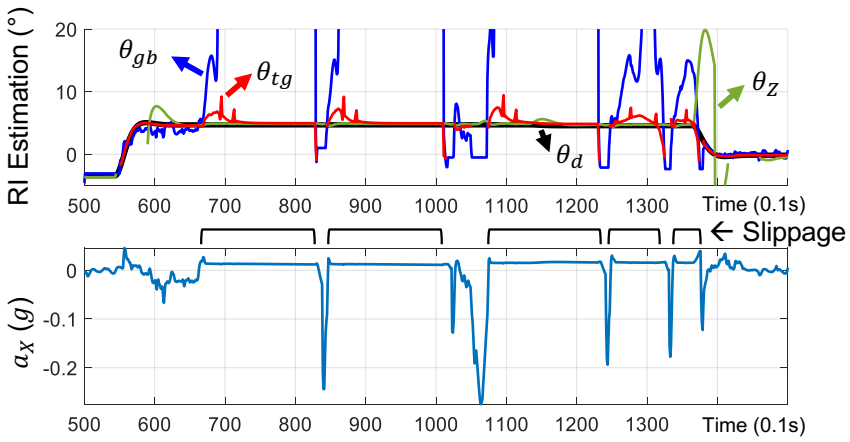


Figure 2.8: Example of tire slippage.

and cargo by checking whether the ratio $(a_X - g\sin\theta_{(d \text{ or } Z)})/(a_X - g\sin\theta_{(gb \text{ or } tg)})$, which is the mass ratio of training time mass to current mass derived from Eq. (2.18), remains at a constant level during a trip. Note that the above filtering mechanism is an optional function since mass change can also be an indication of system tampering.

If RI estimations fail to match TM change, **CADD** further determines if there are signatures indicating tire slippage (Block B2 in Fig. 2.5). Specifically, **CADD** checks if:

- $|\theta_Z - \theta_d| < \eta_{RI}$;
- $\theta_{gb}, \theta_{tg} > (<)$ mean(θ_Z, θ_d) when gas (brake) pedal is applied; and
- a_X remains at the same level.

These signatures are determined based on the fact that tire slippage occurs when the friction between the tires and road surface cannot provide the force required to perform intended acceleration. Therefore, the decrease of acceleration will make θ_{gb}, θ_{tg} to overestimate (underestimate) RI when gas (brake) pedal is applied. In the meantime, $|a_X|$ will be capped at μF_N. Fig. 2.8 is an example of showing the signatures of tire slippage.

2.6.2 Identification of Anomalous DOI

Upon detection of an anomaly, **CADD** further determines whether the anomaly is caused by the DOI related to VB itself or that used to estimate the context. The output

of this identification can be one of the following four possibilities:

S1. a_Z is anomalous;

S2. The direct RI measurement (*i.e.*, θ_d) is anomalous;

S3. Vehicle behavior (VB) is anomalous; or

S4. All DOIs are potentially anomalous.

CADD identifies the source/group of anomaly by finding the outlier(s) in RI estimations. That is, **CADD** will determine whether a set of DOIs is anomalous if the RI estimation based on them deviates from other RI estimations by clustering. Specifically, **CADD** sorts the four estimations based on their values and computes the differences between the sorted values. **CADD** then finds the largest difference between the sorted estimations and divides the group into two subgroups. If there is one group (group A) that only contains one estimation and the largest difference between the estimations in the other group (group B) is less than η_{RI}, **CADD** will report that the DOI used to provide the estimation of group A is the potential anomalous source. For example, the sorted estimations are 1,2,3 and 6°. The differences between the estimations are 1, 1, and 3. Since the largest difference is 3, **CADD** will divide the estimations into two groups, $\{1,2,3\}$ and $\{6\}$. Since the maximum difference of the first group is $3-1=2\leq\eta_{RI}=2$ (*i.e.*, the estimations in this group match each other), **CADD** will determine that the DOI used to estimate 6° is anomalous. **CADD** also determines if the two subgroups are $\{\theta_{gb},\theta_{tg}\}$ and $\{\theta_Z,\theta_d\}$, and if the differences in each group are less than η_{RI}. If yes, **CADD** will conclude that VB is anomalous (S3). If no aforementioned conditions can be found (*i.e.*, RI estimations are not consistent with each other at all), **CADD** will determine that all DOIs are potentially anomalous (S4).

2.7 Evaluation

2.7.1 Experimental Setup

We utilize both i) the data from real-world driving and ii) that generated from CarSim [109] to perform two sets of evaluation. While the first set of evaluation (E_1) provides us with performance evaluation when **CADD** is directly applied to vehicles with limited GPS support, the second set (E_2) shows the true potential of **CADD** when it is applied to advanced vehicles with more accurate GPS/map (or sensors like inclinometer).

2.7.1.1 Generation of Ground Truth Data.

The first set of evaluations (E_1) utilize 13 real-world driving traces we collected by OpenXC VI [98], a commercial off-the-shelf OBD-II dongle. One of these traces is used as training data and the rest as testing data. The training trace consists of 5.1km driving data in an urban area and the testing data include hilly/bumpy roads in freeway, urban, and downtown areas with up to 3.7km (3.8min) traveling distance (time) per trace. The vehicles are driven by 3 different drivers and the roads have up to 5° slope.[3] The vehicle data are directly read off from the CAN bus without any additional information provided from other devices or cloud. Therefore, the road slope (θ_d) required by **CADD** is computed based on the GPS elevation measurements. Note that the elevation measurements from GPS only have coarse-grained accuracy [130]. In the collected traces, the maximum resolution is 3.05m (10ft) and the update rate is ≈1Hz, which can cause up to 14° error to θ_d if the vehicle is traveling at ≥45km/h on a flat road. Therefore, we use 14° as our main threshold setting. Due to safety concerns, we did not manually induce tire slippage for E_1. Instead, we evaluate the scenarios with tire slippage in E_2.

The second set of evaluation (E_2) utilizes data from E_1 with CarSim simulation. We chose CarSim because it can simulate realistic vehicle behavior and road condition (*e.g.*,

[3] In addition to the driver, three traces have no passenger and others have 1 passenger with a total of five ⟨driver, passenger⟩ combinations.

auto-generated pot holes or cracked/bumpy roads), and it is used by 7 major OEMs for design testing [109]. It also allows us to adjust RI and RF without relying on the weather or physical location of testing and, the most important of all, we can test dangerous scenarios (*e.g.*, experiencing excessive tire slippage) without jeopardizing driver safety. We use the driving profiles (with $v \leq 94.5$km/h and $a_X \leq 0.3$g) collected from E_1 to create realistic driving behaviors. That way, our test cases are made close to the vehicle's real-world operations.

We created 10 testing elevation templates. Each template has 10 segments with various RI ranging from -5 to 5° slope (≈7% grades, which is the maximum (urban) RI suggested in the road design guideline specified in Table 8-1 of [6]). The length of a segment projected on the horizontal plane is 250 to 1000m. For each template, we further set the friction coefficient μ to 0.2–0.5, and 0.9, thus creating a total of 50 RI–RF combinations as testing contexts. Together with the driving profile captured in E_1, we use CarSim for VB simulation. To account for the worst-case scenario in which the vehicle has poor sensing quality or is traveling on bumpy roads, we tested the scenarios where there can be excessive measurement (or environmental) noise by injecting AWGN noise with 63% mean and 8% median error to the collected traces. The average length of testing data is 7.5km and the traveling time is 7–8min per trace. The training data consists of 28.25km traveling distance and 41min traveling time on a separate training map (flat road with $\mu = 0.9$).

2.7.1.2 Data Manipulation.

We consider the cases where the vehicle behavior/acceleration (a_X) and assistance data (*i.e.*, θ_d and a_Z) can all be manipulated, spoofed or inaccurate (on top of measurement errors), and see if **CADD** is still able to tell whether the anomaly is caused by the VB related DOI or the incorrect input from other sensors. Table 2.1 lists the manipulation scenarios and thresholds (η_{RI}) we tested for E_1 and E_2. Detection threshold (η_{RI}) also defines the minimum behavior change **CADD** is able to detect ($\Delta a_{X,min}$): $\Delta a_{X,min} \approx g \times \eta_{RI}$. For

Thresholds		E_1: $\eta_{RI} = 7°, 11°, 14°$	E_2: $\eta_{RI} = 1°, 2°, 3°$
Data	Unit	Behavior/Data Deviation	
Long. Acc.	$0.01g$	5, 15, 20, 25, 30	5, 6, 7, 8, 9
GPS Slope	°	8, 10, 12, 14, 16	2, 2.25, 2.5, 2.75, 3
Vert. Acc.	$0.01g$	1, 2, 5, 10, 50	1, 2, 3, 4, 5

Table 2.1: Testing scenarios.

example, $\Delta a_{X,min}$'s of η_{RI} = 7°, 11°, and 14° are 0.12g, 0.19g, and 0.24g, respectively. However, setting a small threshold is not always a good choice since it may induce more false positives. For each scenario, we generate 50 (100) test cases for E_1 (E_2) where the data manipulation starts randomly and lasts for <1min (1–5min), which is equivalent to an average of 63.9% (12.9–64.5%) travelling time. Since **CADD** neither combines consecutive results during detection nor relies on specific attack patterns for its detection, it is the deviation from the normal vehicle behavior (or data value) that matters to **CADD**'s detection, not how the data are manipulated/attacked in time domain. We introduce anomalies by data-manipulation attacks (*i.e.*, modifying the data values to deviate from their original values). Note that a coordinated attack on both VB and an assistance data is equivalent to the other assistance data deviating from its normal value (from a data-level perspective). Since any manipulation in time domain can be considered as a combination of different shifting manipulations, we can directly extend the results of data shifting to other types of attacks by dividing them into smaller segments as individual shifting attacks.

2.7.1.3 Baseline Comparison.

For baseline comparisons, we also implemented EVAD [54] and PID-Piper [28], because, to the best of our knowledge, EVAD is reported to yield the best performance and covers the most similar data types as **CADD** while PID-Piper is the state-of-the-art CI based on machine learning and can be adapted to cover the same detection scope of **CADD**. For a fair comparison, they have access to the same data types as **CADD** does, but they cannot access those detailed data that are not commonly available on in-vehicle networks.

While EVAD [54] has already covered similar data as **CADD**, we implemented the correlation pairs in EVAD that cover the available data listed in Chapter 2.3.1. For PID-Piper [28], its Feed-Forward Controller (FFC) and Feed-back Controller (FBC) are both implemented based on long short term memory (LSTM) machine learning as in [28]. Specifically, the prediction output of FFC is the vehicle's acceleration (a_X) while other available data listed in Chapter 2.3.1 are treated as its input. Similarly, the prediction outputs of FBC are the values of gas (g_a) and brake (b_r) pedals while other available data are treated as FBC's input.

2.7.1.4 Evaluation Metrics.

We use the following metrics for evaluating the *detection* performance of **CADD**.

- *Detection Rate* (*True Positive Rate*): the probability that the anomaly is successfully detected: TPR = TP / (TP + FN), where T (F) is true (false) and P (N) is positive (negative).
- *False Positive Rate*: the probability that **CADD** identifies a normal behavior as an anomaly: FPR = FP / (FP + TN).
- *Detection Latency or Delay* (DL): the time between the anomaly occurs and the time **CADD** determines that an anomaly is taken place. This metric tells us how long it takes for **CADD** to detect an anomaly.

Note that we use *a single data sample* as a unit to compute TPR and FPR, instead of using the entire trip/attack duration.

For **CADD**'s identification performance, we use:

- *Accuracy of Identification* (Acc_{id}): the probability of identifying the exact group of anomalous DOIs.
- *Accuracy of Inclusion* (Acc_{in}): the probability of classifying the anomalous DOI group as potentially anomalous.

Note $Acc_{id} \leq Acc_{in}$. By comparing the two metrics, one can tell whether **CADD** is able to identify the exact group of anomalous DOIs or it just determines all the DOIs as potentially

anomalous all the time.

2.7.2 Performance in Traditional Vehicles (E_1)

2.7.2.1 Detecting VB Change.

Let us consider **CADD**'s performance when it is directly applied to modern vehicles. We use "Δ" in front of a DOI to indicate the amount of manipulation/deviation applied to that DOI/VB. While using $\eta_{RI} = 14°$ and $\leq$1min data manipulation as our main evaluation setting, Fig. 2.9 also shows the receiver operating characteristic (ROC) when η_{RI} is set to 7° and 11°. One can observe that TPRs are always higher than 60% if the manipulation is greater than the minimum detectable value derived by the detection threshold. Even though 60% might seem to be underwhelming, it is actually the result of using each sampling time as one unit to compute TPR. Since **CADD** generates a detection result every detection cycle, the definition of TPR presented here is equivalent to the ratio of the number of cycles **CADD** correctly detects an anomaly to the total number of cycles that a data manipulation is active. If we directly examine whether a single session of data manipulation can be detected before the session ends, there will be 0 false negatives, meaning that each and every manipulation can be captured by **CADD**.

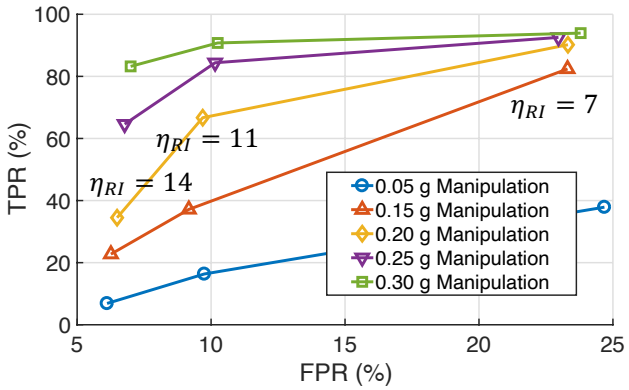


Figure 2.9: ROC curves with a_X manipulation in E_1.

FPR is affected by the choice of detection threshold. **CADD** sets η_{RI} to lower than

the target manipulation level, and higher than the average difference/error between actual road slope and the input slope data. As mentioned in Chapter 2.7.1, since there can be $\geq$14° error when GPS is used to provide θ_d information, we should set η_{RI} to be $\geq$14°. With this setting, **CADD** achieved very low ($\sim$6%) FPR while achieving >80% (>60%) TPR in detecting 0.3g (0.25g) manipulations. We will later discuss how to further reduce false-positives.

2.7.2.2 Detecting Anomalous Assistance Data.

Metric	Δa_Z (g)				
	0.01	0.02	0.05	0.10	0.50
TPR (%)	10.39	27.47	72.82	74.70	74.22
FPR (%)	6.20	6.10	6.49	6.68	6.98
DL (ms)	16204	5666	2495	2311	2759
Metric	$\Delta\theta_d$				
	8°	10°	12°	14°	16°
TPR (%)	8.18	12.94	32.04	54.87	70.40
FPR (%)	6.28	6.56	6.12	6.08	6.17
DL (ms)	17780	14948	9418	7148	6530

Table 2.2: Performance when a_Z and θ_d is manipulated in E_1.

Table 2.2 shows the results when a_Z and θ_d are under data manipulation and η_{RI} is set to 14°. These conditions can also simulate the situation when GPS or accelerometer is spoofed by a malicious party. The TPR remains stable once Δa_Z reaches 0.05g, indicating **CADD**'s capability of capturing data manipulation consistently after this level. As mentioned before, since we only have coarse-grained θ_d input in both temporal and magnitude perspectives and the choice of threshold η_{RI} is bounded by the error level of this input, **CADD** seems to yield lower TPR. However, from the identification results in Table 2.3, **CADD** is shown to be able to achieve an excellent identification rate even if the data manipulation does not reach the theoretical detection level thanks to the low FPR. Specifically, **CADD** is able to achieve 90.67 and 94% rates of anomaly source identification (Acc_{id}) when Δa_X = 0.15g

and $\Delta a_Z = 0.02$g, respectively.[4]

Target	Acc_{id}	Acc_{in}	Acc_{id}	Acc_{in}	Acc_{id}	Acc_{in}
a_X	$\Delta a_X = 0.05$g		$\Delta a_X = 0.15$g		$\Delta a_X = 0.20$g	
	49.33	49.33	90.67	90.67	91.67	91.67
a_Z	$\Delta a_Z = 0.01$g		$\Delta a_Z = 0.02$g		$\Delta a_Z = 0.05$g	
	64.83	64.83	94.00	96.17	96.67	97.67
θ_d	$\Delta\theta_d = 12°$		$\Delta\theta_d = 14°$		$\Delta\theta_d = 16°$	
	30.83	30.83	66.33	66.33	76.80	76.80

Table 2.3: Identification performance in E_1 ($\eta_{RI} = 14°$).

2.7.2.3 Baseline Comparison.

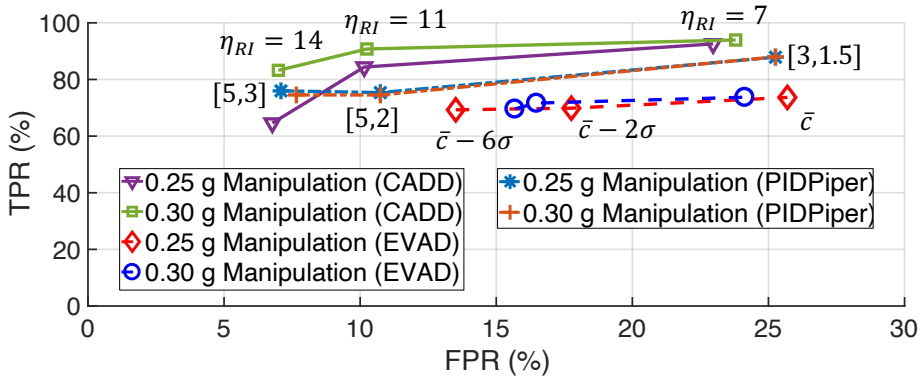


Figure 2.10: ROC comparison between **CADD** and prior studies in E_1, where the labels are the threshold settings of EVAD and PID-Piper, $\bar{c}$ (σ) is the mean (standard deviation) of correlation coefficients of data pairs utilized in EVAD, and the tuple ($[x, y]$) indicates the number of standard deviations to report (x) and cancel (y) an anomaly alarm in PID-Piper.

We now compare the performances of **CADD**, EVAD, and PID-Piper (Fig. 2.10). **CADD** can achieve similar or up to 21.5% higher (absolute) TPR with only a half of EVAD's FPR. The lower TPR and higher FPR of EVAD are the results of EVAD's inability to account for the influence of context when detailed measurements are not available. The results presented here also showcase a major drawback of correlation-coefficient-based approaches: they usually do not have any efficient threshold selection mechanism since correlation coefficients are not directly linked to any cyber-physical properties of the vehicle system.

[4]Modern sedans can achieve >0.5g acceleration (0-60 mph in 3.2s) [132].

That is, we cannot adjust EVAD further to achieve a higher TPR or a lower FPR while maintaining meaningful detection results (*i.e.*, TPR $>$ FPR). On the other hand, PID-Piper is shown to achieve a similar TPR as **CADD** (with a 5–10% difference depending on the settings). We would like to stress again that basic **CADD** *does not* rely on consecutive results for its detection, and hence its TPRs are equivalent to the probability that **CADD** detects an attack based on one single data sample regardless of how attacks/manipulations are launched in time domain. However, because both EVAD and PID-Piper *do* rely on consecutive observations for their detection, their TPRs presented here are the best-case performance when there is a persistent attack lasting for a certain period of time (*i.e.*, larger than their DLs). A further look at the DLs of PID-Piper and EVAD in Table 2.4 reveals that they require more than 7s on average to detect an anomaly while **CADD** incurs $\leq$20% of PID-Piper's/EVAD's DL when considering the settings to achieve low FPR and $\Delta a_X \geq$ 0.25g. These results indicate that a short-lived attack ($<$7s) will have a high probability to evade PID-Pipers's/EVAD's detection and showcase **CADD**'s superiority to both.

Scenario	Δa_X: Amount of Manipulation to a_X				
	0.05g	0.15g	0.20g	0.25g	0.30g
CADD ($7^\circ \approx 0.12$g)	2576	901	417	376	361
EVAD ($\bar{c}$)	12943	10027	11001	10422	10217
PIDPiper ([3,1.5])	1054	1104	1044	1102	1155
CADD ($11^\circ \approx 0.19$g)	6282	4272	1652	977	559
EVAD ($\bar{c} - 2\sigma$)	18664	11829	11626	12457	11581
PIDPiper ([5,2])	7242	7225	7219	7207	7193
CADD ($14^\circ \approx 0.24$g)	18076	6573	5571	2171	1628
EVAD ($\bar{c} - 6\sigma$)	19068	13503	12118	12613	12730
PIDPiper ([5,3])	7254	7212	7213	7208	7194

Table 2.4: Detection delay (ms) in E_1. Note that some test cases do not exceed the detection threshold (*i.e.*, η_{RI} or $\Delta a_{X,min}$).

2.7.3 Performance with Map Support (E_2)

We now explore **CADD**'s performance when more accurate θ_d (*i.e.*, inclination estimation) support is available. As mentioned in Chapter 2.3.1 and Fig. 2.5, θ_d does not have

to be obtained from the elevation data of GPS (as in E_1). That is, it can also come/computed from other available sensor data, such as i) the combination of GPS and Map, ii) an inclinometer, or iii) LIDAR/camera as long as the inclination is not computed based on a subset of other data groups in Fig. 2.5. For example, Wang *et al.* [142] showed that Google Earth can achieve 1.32m mean absolute error in the US while Khalid *et al.* [33] showed that Google Earth can achieve as low as 0.51m mean error in some specific regions (*e.g.*, Dabaa City, Egypt), which can be translated to 2.7 and 1.05° error when the vehicle is traveling at 100km/h and 1Hz update rate, respectively. Commercial inclinometers are claimed to achieve 0.1–2° error [107, 111, 112] and slope estimation based on vision sensors (*e.g.*, camera) is shown to achieve 0.2° error [139]. Even though the above technologies may not be available to every car and everywhere on earth, they indicate that a more accurate θ_d (than E_1) can be obtained for **CADD** to realize its true potential in future.

2.7.3.1 Without Tire Slippage.

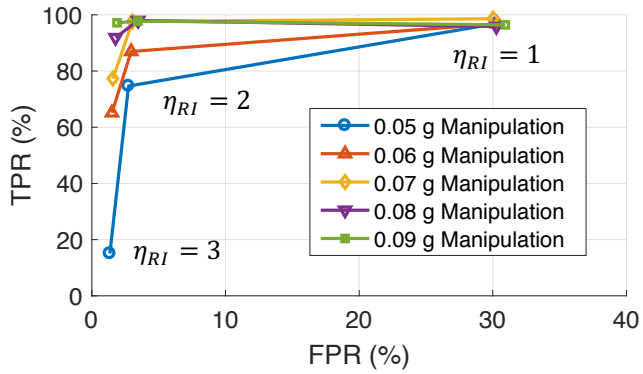


Figure 2.11: ROC curves without tire slippage in E_2.

In E_2, we use $\eta_{RI} = 2°$ as our target evaluation scenario based on the noise level observed in the training data. Fig. 2.11 shows ROC curves when η_{RI} is set to 1 – 3° (marked on each curve) and the curves indicate different levels of behavior deviation. **CADD** is able to achieve a high TPR (75%) even if the deviation (Δa_X) is only 0.05g and its FPR is only 4% ($\eta_{RI} = 2°$). Note that we injected a large amount of noise (*i.e.*, 13% of data have

larger than 50% noise) in our test cases, which will inevitably generate false positive detection. **CADD** can efficiently detect any behavior deviation greater than 0.07g with 97% TPR while the mean DL is less than 1s (Table 2.5).

η_{RI}	Δa_X: Amount of Manipulation to a_X				
	0.05g	0.06g	0.07g	0.08g	0.09g
1°	524	396	409	414	325
2°	1905	1034	802	745	620
3°	11597	2642	1908	1175	957

Table 2.5: Detection delay (ms) without tire slippage in E_2.

Developers can also plot the ROC curve (from the training data) as a guideline for setting η_{RI}. For example, if we assume Fig. 2.11 were the ROC obtained from training phase and a developer wishes to detect any behavior deviating from its norm by 0.06g with $>$60% TPR and $<$2% FPR, then, according to the ROC curves, s/he should set η_{RI} to 3°.

2.7.3.2 In the Presence of Tire Slippage.

Let us consider **CADD**'s detection performance in the presence of tire slippage (TS), simulating the vehicle traveling under different weather conditions that can cause different levels of road friction. Table 2.6 shows **CADD**'s performance when μ is set to 0.2–0.5 and 0.9, where 0.9 is the road condition we used in Chapter 2.7.3.1. The average time of TS (in percentage of the entire trip) is also listed in Table 2.6. We can observe that **CADD**'s performance is almost identical to the condition without TS when there is no excessive TS during the trip (*i.e.*, μ = 0.3–0.5). Even when there is up to 34% of TS time (μ = 0.2), **CADD** is able to achieve $\sim$92% of TPR and only $\sim$7% of FPR when the Δa_X = 0.07g. That is, **CADD** is able to significantly reduce the occurrence of false-positive detections (from potentially up to 34%) to only 7%. The slightly lower TPR than the scenario without TS (μ = 0.9) is due to TS's cancellation of the effect of an a_X change. When a_Z or θ_d is anomalous (Table 2.7)[5], **CADD** is able to achieve almost identical TPRs and FPRs with or

[5]We omit the results when μ= 0.3–0.5 in these two tables because they are almost identical to μ=0.9 as in Table 2.6.

Metric	Δa_X: Amount of Manipulation to a_X				
	0.05g	0.06g	0.07g	0.08g	0.09g
$\mu = 0.9$ (No Tire Slippage)					
TPR (%)	75.05	87.65	97.03	96.59	97.83
FPR (%)	4.02	4.24	4.53	4.60	4.67
DL (ms)	1905	1034	802	745	620
$\mu = 0.5$ ($< 1\%$ Slippage Time)					
TPR (%)	74.90	87.63	96.98	96.36	96.12
FPR (%)	4.07	4.31	4.48	4.75	4.72
DL (ms)	1680	1124	895	746	771
$\mu = 0.4$ ($< 1\%$ Slippage Time)					
TPR (%)	75.02	87.50	97.94	97.77	96.96
FPR (%)	4.09	4.34	4.61	4.61	4.78
DL (ms)	1663	1168	799	764	613
$\mu = 0.3$ ($< 1\%$ Slippage Time)					
TPR (%)	75.11	87.48	97.07	96.97	96.68
FPR (%)	4.28	4.47	4.70	4.79	5.06
DL (ms)	1608	1032	863	712	786
$\mu = 0.2$ ($5 \sim 34\%$ Slippage Time)					
TPR (%)	68.27	80.39	91.88	93.49	94.09
FPR (%)	6.08	6.29	6.63	6.69	6.85
DL (ms)	2027	1663	1315	1226	998

Table 2.6: Detection performance in E_2 (a_X anomalous).

without TS. That is, **CADD** is able to distinguish TS from the actual anomaly and output the correct results. Note that since **CADD** does not combine consecutive results for its detection, the performance of traveling on a road with changing μ can be approximated by considering small road segments with different μ.

Next, we compare **CADD**'s FPR performance with EVAD by adjusting EVAD's threshold settings to have the same level of TPR as **CADD** (Fig. 2.12). Similar to the results shown in E_1, **CADD**'s FPR is only 34.5–38.5% of that of EVAD regardless of (non-)occurrences of tire slippage. For PID-Piper, it can only achieve $<$30.7% TPR when having a similar FPR ($\sim$4%) as **CADD** (Fig. 2.12). If we adjust PID-Piper's settings to achieve similar TPR ($>$90% when $\Delta a_X \geq 0.07$g) as **CADD**, its FPR increases to $>$90%. Even for given special training data with 10 different RI conditions, PID-Piper still suffers a 66.95% FPR. This result further shows a major drawback of a ML-based approach without physical

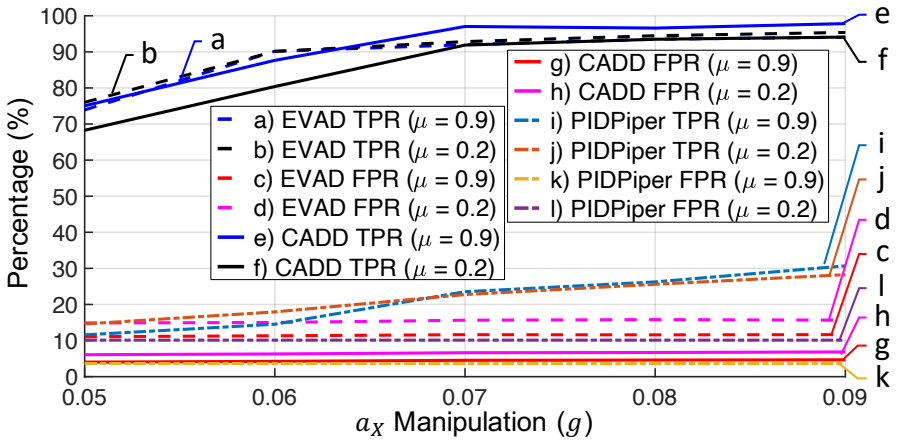


Figure 2.12: Comparison between **CADD** and prior work in E_2.

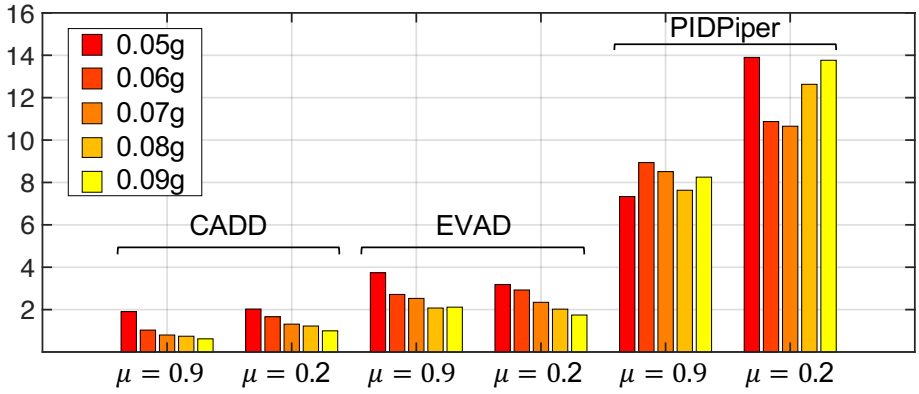


Figure 2.13: Detection latency (seconds) of **CADD** and prior work when $\Delta a_X = 0.05 \sim 0.09$g in E_2.

modeling: its detection will be ineffective when operating in a constantly changing environment because of the training difficulty and the lack of necessary data access. On the other hand, since **CADD**'s and EVAD's detection is based on the vehicle's physical model and data correlation, there will be no significant performance degradation under such a condition. Furthermore, **CADD** is shown to achieve <51% of EVAD's/PID-Piper's DL thanks to **CADD**'s model-based detection mechanism (Fig. 2.13).

	Δa_Z (0.01g)					$\Delta\theta_d$ (°)				
	1	2	3	4	5	2	2.25	2.5	2.75	3
$\mu = 0.9$ (No Tire Slippage)										
TPR	87.8	88.6	91.2	92.7	92.7	78.7	95.5	98.0	98.6	99.1
FPR	4.5	4.7	4.7	4.8	4.7	3.6	3.5	3.6	3.7	3.6
DL	7.38	7.54	7.21	7.41	7.44	0.79	0.26	0.07	0.08	0.02
$\mu = 0.2$ ($5 \sim 34\%$ Slippage Time)										
TPR	88.3	89.1	91.1	93.1	93.3	79.1	96.8	98.4	98.9	99.0
FPR	7.0	7.0	7.1	7.1	7.2	5.7	5.7	5.8	5.8	5.9
DL	7.58	7.16	7.32	7.71	7.47	0.57	0.15	0.09	0.07	0.06

Table 2.7: Detection results of E_2 (a_Z or θ_d anomalous), where TPR, FPR, and DL are presented in %, %, and s, respectively.

2.7.3.3 Identifying an Anomalous Group

We now look at **CADD**'s end-to-end performance in not only detecting the occurrence of an anomaly but also pinpointing the anomalous group. First, we consider the condition in which a_X is anomalous as in Chapter 2.7.3.1. We then test the conditions in which a_Z and θ_d are anomalous, simulating the scenario that the sensor data used for context estimations are faulty. The purpose of this evaluation is to see whether **CADD** can correctly identify the group of anomalous DOIs. In this evaluation, we set **CADD** to output an identification result when each trip ends. This identification result is determined by taking the majority of anomalous groups captured in the identification process (*i.e.*, S1–S4 in Chapter 2.6.2) within a single trip. Table 2.8 summarizes the results of identifying an anomalous group. Since the results of μ = 0.3–0.9 are, in general, identical, we only present the results of μ = 0.9 and 0.2 here. The "Ano" column is the anomalous DOI, and the "AD" column is the anomalous duration in minutes. We tested the cases with 1, 2.5, or 5min AD, which translate to 12.9, 32.3, and 64.5% anomalous duration ratio (ADR), respectively. The second row (Dev. Lvl.) of the table indicates the level of deviation (k) from the ground truth. The actual deviation is given by Δa_X = [0.04 + 0.01k]g, Δa_Z = [0.01k]g, and $\Delta\theta_d$ = [1.75 + 0.25k]°. While $|Acc_{in} - Acc_{id}| < 0.1$ holds for all the scenarios we have tested, indicating the rare occasion of **CADD**'s inability to determine an anomalous DOI, here we

only show the values of Acc_{id}.

Ano | AD		$\mu = 0.9$					$\mu = 0.2$				
Dev. Lvl.		1	2	3	4	5	1	2	3	4	5
a_X	5	100	100	100	100	100	97.5	99.5	99.5	100	100
	2.5	94.4	97.0	97.7	99.3	99.1	89.1	95.2	96.8	97.1	99.5
	1	90.8	96.5	98.8	98.5	99.0	97.5	98.0	99.5	100	99.5
a_Z	5	100	100	100	100	100	98.9	100	100	100	100
	2.5	100	100	100	100	100	91.8	92.2	93.2	93.0	90.0
	1	97.5	95.8	98.5	94.8	96.0	62.7	62.5	67.5	68.2	62.0
θ_d	5	100	100	100	100	99.9	95.0	99.2	99.4	98.9	98.2
	2.5	93.4	96.5	94.5	93.1	90.7	75.7	85.8	85.4	82.5	81.3
	1	81.2	85.5	84.0	80.2	79.5	55.4	63.9	67.0	63.5	62.7

Table 2.8: **CADD**'s source identification results (Acc_{id}) in E_2. See Chapter 2.7.3.3 for detailed description.

When μ = 0.9 (*i.e.*, without TS) and a_X or a_Z is anomalous, **CADD** is able to identify the exact source of anomaly with $\geq$94.4% Acc_{id} if $ADR \geq$ 32.3%. Even if ADR = 12.9%, **CADD** can still achieve $\geq$90.8% Acc_{id}. These results show **CADD** to be able to identify the anomalous source even when the anomaly lasts only for 1 min (ADR = 12.9%). Compared to the condition without TS, **CADD** has lower Acc_{id}, especially in view of the performance when a_Z or θ_d is anomalous. The lower Acc_{id} is the result of **CADD**'s tendency to determine the source of anomaly as a_X when TS occurs. That is, **CADD** won't be able to distinguish whether the anomaly is caused by anomalous a_X or the other two DOIs, and it will determine the source to be a_X because $\Delta\theta_{gb}$ and $\Delta\theta_{tg}$ caused by TS will be larger than $\Delta\theta_d$ in our testing scenarios, where $\Delta\theta_X$ is the deviation of estimation θ_X from the ground truth. Nevertheless, **CADD** is still able to achieve $\geq$75.7% Acc_{id} when $ADR \geq$ 32.3% and $\Delta\theta_d = 2°$.

2.7.4 Remark

2.7.4.1 Attack Stealthiness

Note the attacks considered here are already stealthy (or small enough) in that i) their manipulation levels are within the common permissible error and ii) they can evade detec-

Data	Unit	Magnitude of Manipulation or Deviation								
Δa_X	10^{-2}g	5	6	7	8	9	15	20	25	30
Δv	10^{-2}km/h	18	21	25	28	32	53	71	88	106
ΔX	10^{-4}m	25	29	34	39	44	74	98	123	147

Table 2.9: This table shows the resulting effects (per data sample taken at a 10Hz rate) of data-manipulation attacks, where Δa_X is the attack magnitude, Δv and ΔX are the resulting (maximum) speed and location deviation/errors perceived by the vehicle based on physical modeling.

tion by prior approaches with >7s DL in E_1 (Δa_X = 0.3g) or >1.7s in E_2 (Δa_X = 0.09g). Table 2.9 shows the speed and location deviation under different levels of data manipulation tested in this section. Specifically, even if we look at the maximum manipulation (*i.e.*, Δa_X = 0.3g), the attack will only generate a 0.0147m location deviation and a 1.06 km/h speed deviation every 0.1s while pure GPS localization is known to have meter-level error (4.9m [52]) and the European law (ECE-R39) only requires the speedometer to report with less than $0.1v_{GT}$ + 4 km/h error, where v_{GT} is the ground truth of the vehicle speed. Note that **CADD** is shown to require only one anomalous data sample to detect an anomaly in >70% test cases as shown in Fig. 2.14.

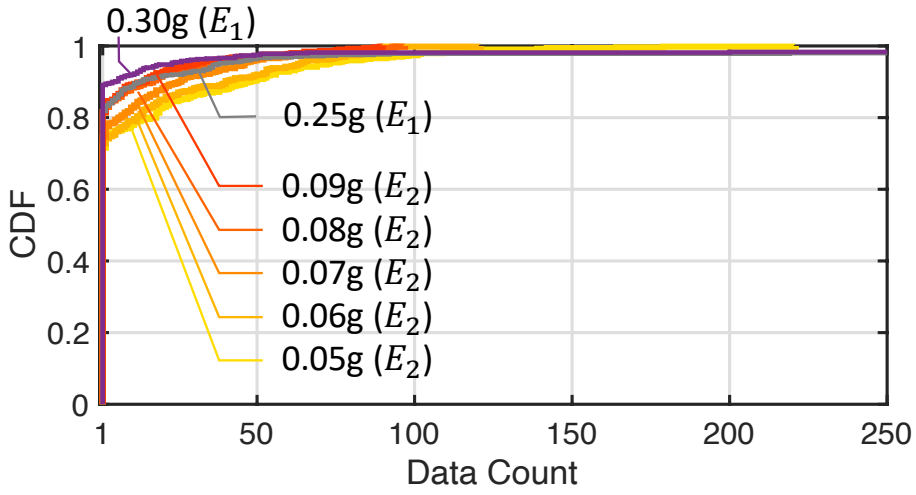


Figure 2.14: CDF of the required data count for CADD to detect an anomaly (as an alternative way to present detection latency) when Δa_X is anomalous.

ADR	Metric	μ = 0.9, FPR = 4.41%			μ = 0.2, FPR = 6.51%		
Manipulation		0.01g	0.02g	0.03g	0.01g	0.02g	0.03g
64.5	TPR	4.27	8.62	20.57	7.58	10.05	20.65
	Acc_{id}	71	92	100	90	96	100
32.3	TPR	4.27	8.43	19.93	7.14	10	20.29
	Acc_{id}	70.6	81.6	99.4	90	93.8	99.8
12.9	TPR	4.29	7.77	18.93	7.05	9.83	19.87
	Acc_{id}	71.8	76.6	95	90	91.2	96

Table 2.10: This table shows the detection and identification performance (%) of **CADD** when Δa_X is smaller than the detection threshold ($\Delta a_{X,min}$ = 0.035g).

2.7.4.2 Resilience against Different Attacks

Let us consider the case when the attacker knows the parameter settings in **CADD** and tries to evade **CADD** by manipulating data within the thresholds or manipulating multiple data at the same time. Table 2.10 shows **CADD**'s detection and identification performance when Δa_X is smaller than the detection threshold (η_{RI} = 2 or $\Delta a_{X,min}$ = 0.035g). As expected, **CADD** can only achieve moderate TPR since the data manipulation is below the detection threshold. However, **CADD** is shown to achieve $\geq$71.8% Acc_{id} even if ADR is only 12.9%. These results indicate that even with data manipulation equal to only 28.6% of detection threshold, **CADD** can still capture/pinpoint the data in question with good accuracy. However, any manipulation under the detection threshold should not be considered as a valid attack since, in a practical deployment scenario, the detection threshold should also account for the error tolerance of the vehicle system.

As described in Chapter 2.3, a natural limitation of **CADD** is that it cannot detect a full-scale data manipulation, which is a common characteristic of all approaches without the data of final control output/setpoint (or a trusted data source). However, when map support is also available to the vehicle in the future, a real-time full-scale attack will be very difficult, if not impossible, to evade **CADD**'s detection for the following reasons.

To generate a full-scale attack in real time, it requires the attacker to change the road inclination context θ_d utilized in **CADD** to match DOI after manipulation. With map sup-

port, θ_d will not be directly computed based on the elevation measurements embedded in the GPS data on IVN. Instead, **CADD** will use the geo-coordinates to look up θ_d on the map, meaning that the attacker will need to find a series of geo-coordinates with exactly same θ_d's that match the manipulated DOI (if such series of geo-coordinates even exists). Even if θ_d is obtained from other estimation methods as described in Chapter 2.7.3, map can be used as a source for sanity check. Since any value gap between two consecutive GPS data is also an indication of a data manipulation, launching such an attack is infeasible. On the other hand, to launch a replay attack or a pre-computed attack, unless *every* data in the beginning of the recorded trace used by the attack exactly matches *every* DOI of **CADD** when the attack is launched, there will be value gaps between before and after the replay attack is launched, thus leaving an obvious signature of data manipulation.

Based on the above discussion, we also tested the condition where all dynamics measurements and the assistance data, except for θ_d, are manipulated simultaneously. Note that the evaluation settings here are basically the same as in E_2, except that all aforementioned data are manipulated simultaneously to match the normal (physical) data correlation (*e.g.*, vehicle acceleration now perfectly matches DOIs other than θ_d while ignoring measurement noise) and the vehicle behavior tested in the test cases are indeed captured from the normal vehicle behavior but in a different driving context (*i.e.*, RI). As shown in Fig. 2.15, **CADD**'s TPRs are almost identical to those shown in Fig. 2.12. EVAD and PID-Piper, however, can only achieve $<$20% TPR because the former does not have any special design to consider driving context and the latter cannot efficiently account for θ_d to detect the anomaly while other data match their normal correlation. As a result, they also have significantly higher detection delays than **CADD** (Table 2.11).

We now consider the condition in which the adversary tries to evade **CADD**'s detection by disguising the attack as a mass change to the vehicle. Note any vehicle behavior change caused by total mass (including passengers and cargoes) change must be consistent throughout the entire trip as described in Chapter 6.1. Therefore, in order for an attack

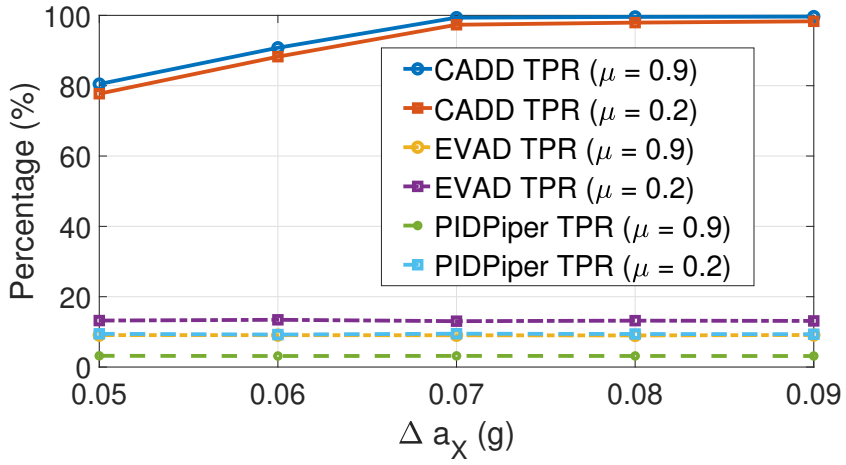


Figure 2.15: Comparison of TPRs between **CADD** and prior work under (coordinated) multi-data manipulation attacks. Since the FPRs will not be influenced by the attacks, the FPR comparison is the same as Fig. 2.12.

Scenario (μ)	0.05g	0.06g	0.07g	0.08g	0.09g
CADD (0.9)	870.0	238.0	8.8	7.4	8.4
CADD (0.2)	849.8	212.4	16.6	25.4	70.0
EVAD (0.9)	58707.8	56852.2	57956.2	56511.0	57471.6
EVAD (0.2)	44644.6	46727.8	46164.4	41862.8	46385.4
PID-Piper (0.9)	13547.8	16276.6	16913.2	14846.4	16472.0
PID-Piper (0.2)	13771.8	17128.4	16763.8	15688.4	16159.4

Table 2.11: This table shows the detection latency (ms) of **CADD** and prior work under (coordinated) multi-data attacks.

to disguise as a mass change, the manipulated acceleration must be maintained at a constant ratio, compared to the ground-truth acceleration, throughout the entire trip. This will further lead to the requirement of simultaneously manipulating vehicle speed and GPS to evade a simple consistency check between those sensor readings. However, the road grade information from a map (looked up by using GPS readings) cannot be controlled by the attacker and, therefore, the inconsistency between road grade and vehicle acceleration can be detected by **CADD**

Also, for an attack to disguise as a tire slippage by manipulating both control input and longitudinal acceleration, the adversary needs to further manipulate both road grade estimation (*e.g.*, GPS or GPS and map) and vertical acceleration readings simultaneously

to match proper tire slippage features w.r.t. to the road grade estimation obtained from gas/brake and engine torque. This will lead to manipulating all the status measurements considered in **CADD**, which is outside of **CADD**'s detection scope.

2.7.4.3 False-Positives

In general, a detection system may have false-positives because of i) imperfect modeling caused by limited data availability and accuracy or ii) transient noise in sensor readings that exceeds the detection threshold. While **CADD** has already considered the former and utilizes the threshold setting that is above the average measurement error level (Chapter 2.7.1), most reported false-positives are caused by transient noise. Based on this observation, **CADD**'s FPR can be reduced further by a false-positive filtering mechanism (FPFM) even though **CADD** can already achieve low FPR under excessive noisy conditions. Specifically, since any effective attack must last for a certain period of time to pass through a low pass filter (*i.e.*, a common practice of data/signal processing to reduce the effect of noise), **CADD** can choose to report an anomaly detected only when there are α consecutive positive detections, where α is a design parameter to balance between the FPR and the detection delay.

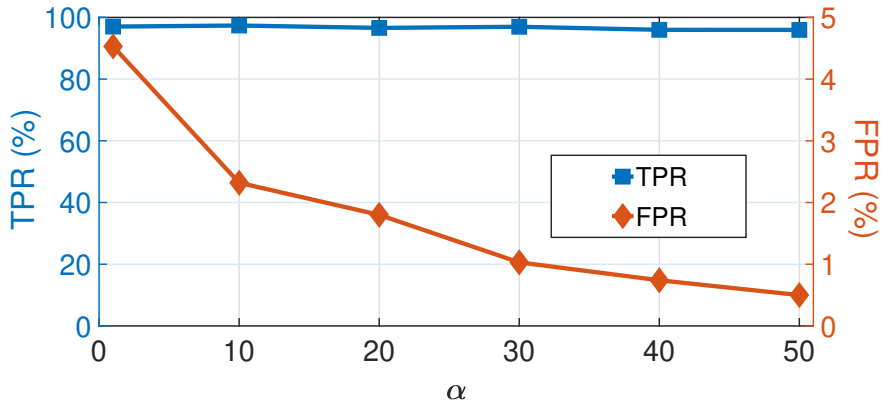


Figure 2.16: **CADD**'s performance after applying FPFM.

Fig. 2.16 shows **CADD**'s detection performance when $\Delta a_X = 0.07g$ under different α values. We can observe that FPFM can effectively reduce FPR (from 4.5% to 0.5%) when

$\alpha = 1 \rightarrow 50$ detection cycles. Furthermore, this makes no significant impact on overall TPR (from 97.0% to 96.0%), proving its effectiveness in enhancing **CADD**'s performance. Since FPR is independent of the manipulation/attack level, there is no need to adjust α according to the target detection level. However, a larger α incurs a larger detection delay while providing a better FPR performance. Specifically, Fig. 2.17 showcases **CADD**'s detection latency with FPFM applied. The results have shown **CADD** to successfully report an anomaly right after the observation window of FPFM in 50–80% (>40%) test cases even when α = 1–40 (α = 50), indicating **CADD** can still capture anomalies immediately most of the time (α = 1–40) right after their occurrences (but just chooses to report them at a later time).

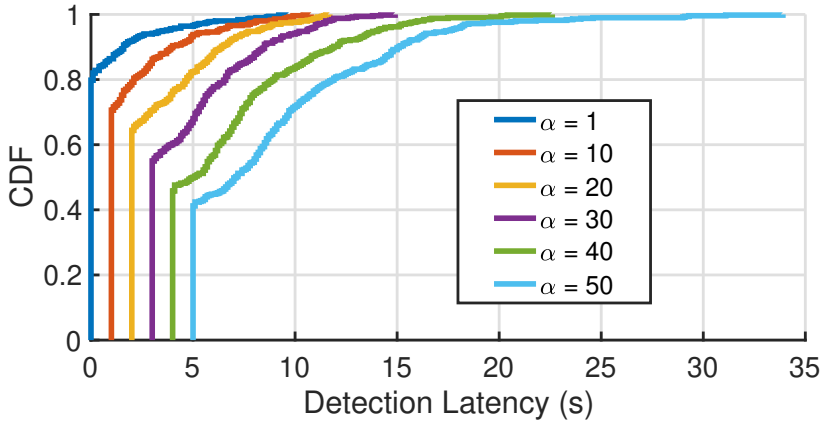


Figure 2.17: CDFs of **CADD**'s detection latency with FPFM.

If **CADD** is implemented as an alarm system, detection delay will not be a major concern, and the application developers should provide the flexibility to users to adjust how aggressive FPFM should be according to their preference. On the other hand, if **CADD** is to be used for on-line defense that directly integrates into the vehicle's control system, the value of α should also depend on the system requirement/characteristics, including the maximum tolerable detection delay and the maximum tolerable attack duration, which are known to the component suppliers or car makers. In other words, even if the attacker tries to evade **CADD**'s detection when FPFM is applied by shortening its attack duration, the

system should be able to handle the absence of correct data during that period of time.

2.8 Discussion

2.8.1 Deployment and Application of CADD

CADD is designed with easy deployment in mind. Since all the required data can be obtained entirely from the vehicle itself, or from the combination of the vehicle and an external device equipped with GPS/INS (*e.g.*, a smartphone) inside the vehicle, **CADD** can be implemented in a module attached to the CAN bus in the vehicle or as a dongle plugged in the OBD-II port. While the former is targeted at the first-party implementation as a built-in function in the vehicle, the latter is likely a third-party add-on implementation. As a first-party, there will be no problem retrieving all the necessary data from the vehicle's internal components and, therefore, this implementation does not need any extra sensors to be installed. On the other hand, while implemented as a third-party solution, the data required by **CADD** may not be accessible from the standard OBD-II messages (*e.g.*, acceleration). Therefore, an extra IMU is required in the implementation. While insurance companies want to prevent accidents to increase their profits, they will have incentives to deploy **CADD** to alert their customers about potential anomalies. Furthermore, since most of the OBD-II dongles provided by insurance companies already have built-in IMUs, they will not incur additional hardware cost to add **CADD** in their dongle.

There are two aspects of safety enhancement, especially in the cyber-physical system (CPS) domain like cars, trains and planes. The first is an early warning and suggestion of preventive measures. An anomaly detection system will try to identify any early sign of anomaly and warn/advise the users to perform further inspection in order to prevent a more severe, safety-critical situation in the near future. This type of safety enhancement does not necessarily need to be done in real time. The second aspect is on-line protection dealing with a more strict timing constraint. While **CADD** can achieve $<$100ms detection delay, it

can be utilized as a on-line protection system if FPFM is set to use a small α. On the other hand, the application of FPFM with a larger α will be more leaning towards preventive safety enhancement.

The other potential application of **CADD** is to detect modifications to driving traces for an application (*e.g.*, usage-based insurance as mentioned earlier). For example, while relying only on the IMU of a OBD-II dongle to detect reckless driving can be easily deceived by sensor spoofing (*e.g.*, using a speaker [96]), the combination of in-vehicle data and the IMU readings will raise the level of difficulty to deceive the application.

While there will never be a perfect detection system that can capture all types of attack, **CADD**'s limitation is that it cannot detect a full-scaled manipulation attack as discussed earlier. However, **CADD** is able to significantly raise the bar to launch a successful attack. Furthermore, it can also be utilized as a general fault detection system to detect early signs of component failure.

2.8.2 Consideration of Other Context (Σ)

2.8.2.1 Steering

As part of our future work, we discuss how to improve our model formulation if more data types become available to **CADD** in the future. First, we look at the adjustments that can be influenced by vehicle steering (*i.e.*, $S_{E(\Sigma)}$ and $S_{B(\Sigma)}$ in Eq. (2.3)). Specifically, since the force generated by wheel torque will now contribute to both longitudinal and lateral acceleration when the vehicle has lateral movement, $S_{E(\Sigma)}$ and $S_{B(\Sigma)}$ will be dependent on the steering angle (*i.e.*, the angle ϕ_T between the tire direction and the vehicle's body direction) which is determined by the *steering wheel* angle (ϕ_s) and its corresponding steering ratio $k(\phi_s)$ [77]:

$$\phi_T = \phi_s / k(\phi_s). \tag{2.19}$$

$S_{E(\Sigma)}$ ($S_{B(\Sigma)}$) can then be presented as a function of ϕ_s as $S_{E(\Sigma)}(\phi_s)$ ($S_{B(\Sigma)}(\phi_s)$). The reason for utilizing ϕ_s instead of ϕ_T is: i) the former is more commonly available on IVN in drive-by-wire vehicles and ii) modern vehicles usually have different ϕ_T's between tires to maintain a consistent center of turning circle (O) as shown in Fig. 2.18 [68]. Note that we preserve (Σ) in the presentation since there are still other factors that can influence S_E, such as sideslip caused by the deformation of the tires. For a generic steering design, **CADD** can utilize the training data when the vehicle is on a straight road to obtain $\mathcal{M}_{gb}$ and $\mathcal{M}_{tg}$ without the adjustment of $S_{E(\Sigma)}$ and $S_{B(\Sigma)}$, and then performs training with data when $\phi_s \neq 0$ to obtain $S_{E(\Sigma)}$ and $S_{B(\Sigma)}$. However, to account for a more sophisticated steering assistance system, **CADD** can utilize a similar training procedure of $\mathcal{M}_{gb}$ as described in Chapter 2.5.2 that partitions the training data according to both v and ϕ_s (instead of just v) to capture the steering adjustment in different driving scenarios.

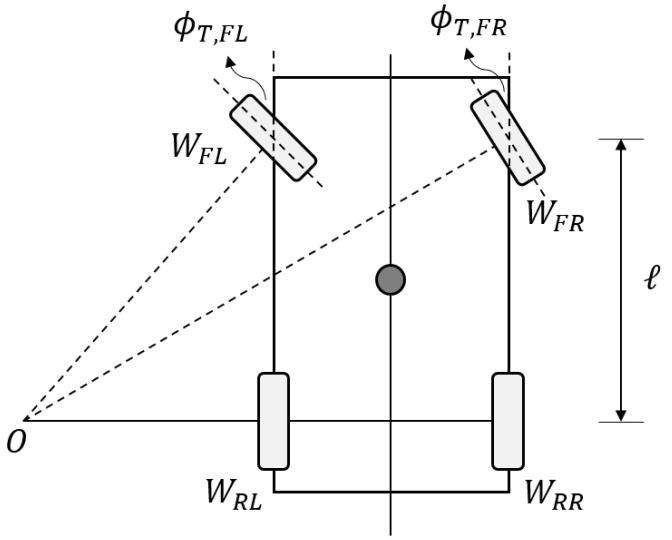


Figure 2.18: Example of steering angles (Ackermann steering geometry [68]).

2.8.2.2 Fuel Consumption and Mass Change

While fuel level is *not* a common data that can be easily retrieved from IVN, **CADD** by default treats the mass change due to fuel consumption as a model noise. This assumption is grounded on the fact that modern (medium size) vehicles usually weigh more than 3000lbs

(1360.7kg) [95] and the average fuel consumption is 30.8mpg (miles per gallon) [138]. That is, there will be only 0.2% mass change for every 30.8 miles (49.57km) of travel. However, if fuel-level information is available, it can be accounted for by Mass Change Detector.

2.9 Conclusions

We have presented **CADD**, an anomaly detection system that utilizes multiple sets of data for detection of context information (*i.e.*, road inclination, tire slippage and total mass) and determines whether an anomaly has occurred by comparing the context estimations. It not only detects the occurrence of an anomaly but also narrows down the search space of the anomaly source by comparing the estimations and excluding the group of data that match each other from the search space. Our extensive evaluation (87,000+ test cases, including trace- and simulation-based evaluations) has shown **CADD** to achieve high (>96%) detection rate (TPR) and low false positive rate (<0.5%) even in the presence of excessive measurement noise. **CADD** can also identify the anomaly source with accuracy of >95% even if the anomalous duration is only 32.3% of the observation time ($\Delta a_X = 0.07g$).

CHAPTER III

DiVa: Identification of Anomalous Source(s)

3.1 Introduction

SA systems, such as commercial vehicles, are getting equipped with an increasing number of sensors and communication interfaces to provide advanced services like advanced driver-assistance systems or autonomous driving. Car makers are also gradually replacing the traditional mechanical components with their electronic counterparts for vehicle control. While the increase of electronic components and communication interfaces enables advanced services [83], it also comes with reliability/security concerns. Unsafe situations can occur due to faulty hardware or buggy software components (*e.g.*, Toyota's unintended acceleration [154]) or even malicious attacks (*e.g.*, Jeep Cherokee hack [87]).

Considering the safety risks associated with anomalous system behavior, researchers have been exploring how to automatically determine if the behavior of a target system deviates from its normal operation. To detect attacks initiated from in-vehicle networks (IVNs), researchers have also been developing intrusion detection systems (IDSes) for vehicles [3, 22, 92, 93, 145]. Since every electronic control unit (ECU) in a vehicle can only transmit a fixed set of messages, such IDSes are designed to detect the messages transmitted by rogue ECUs by exploiting the data transmission characteristics, such as transmission rates [92, 93] or voltage signatures of ECUs [22]. They can achieve good detection performance, but cannot detect the anomalous vehicle behavior that does not alter the transmis-

sion patterns/characteristics on IVNs and cannot be applied to other types of systems with different network configurations.

To detect the conditions (*e.g.*, pure data or behavior anomalies) the aforementioned IDSes cannot cover, various data-driven schemes [2, 5, 22, 23, 45, 50, 54, 79, 90, 92, 93, 101, 102, 124, 146, 153, 157] have been proposed to identify general abnormal system behavior instead of focusing only on abnormal data-transmission patterns on in-system/vehicle networks. However, there still does not exist any practical data-driven approach that can *simultaneously* achieve i) good performance, ii) anomalous source identification, and iii) easy deployability without requiring unavailable or difficult-to-access data.

We will use an SA vehicle as a concrete example. Specifically, while the standardized on-board diagnostic II (OBD-II) messages based on SAE J1979 cover only the data types related to transmission, and the controller area network (CAN), the *de facto* standard IVN, has only limited bandwidth.[1] The commonly available data on IVNs are limited to certain control inputs, powertrain component states and dynamic measurements. Note that the controller setpoints and the measurements of actuator outputs (*i.e.*, wheel/brake torque), which are necessary inputs for most prior work, are *not* commonly available on IVNs (see Chapter 3.2). This unrealism/limitation of data availability is also the very reason why existing detection schemes, including control/system invariant approaches, cannot be directly applied to real SA systems without incorporating proper system models from control inputs to setpoints or dynamics while taking practicality and deployability into account.

As the human's/driver's control and monitoring has been decreasing or even disappearing, automatic detection *and* identification of the source(s) of anomalies (*i.e.*, attacks and faults) become critically important. An SA system must be able to automatically perform diagnostic and report the results to system owners/managers for further inspection because early discovery of anomaly and location of its source can prevent the occurrence of safety-critical situations caused by the same vulnerability exploitation in the future. The

[1] 1Mbps with a high-speed CAN [62] and 5–12Mbps with a CAN-FD [61].

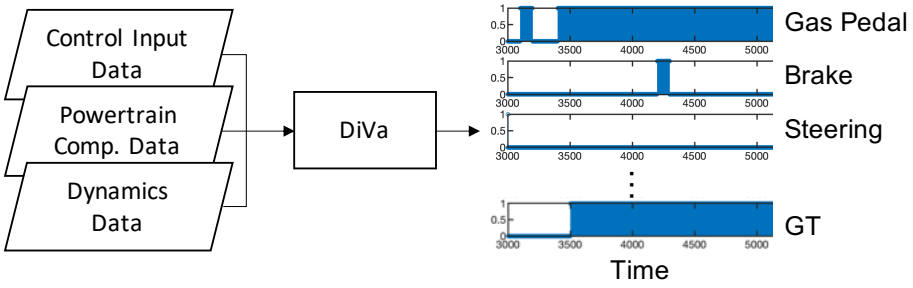


Figure 3.1: This figure depicts **DiVa**'s functional overview and an example of **DiVa**'s output (using a vehicle as a concrete example), where the colored blocks indicate the time when the data is determined to be a potential anomaly source and the last row indicates the ground truth (GT) when the gas pedal is the anomaly source.

identification of anomalous data is important for SA systems to "signal" how serious the condition may be, depending on the anomaly source. Without this capability, the SA system can only issue an alarm and inform drivers or mechanics to conduct a full inspection of the car, which can be time-consuming and costly.

To address the lack of a practical approach that focuses on the forensics of SA anomalies, we propose a novel approach, called **DiVa**. The goal of **DiVa** is to provide a function akin to a root cause analysis that can be used to identify (or track down) the source of anomaly. Specifically, **DiVa** is designed to detect and narrow down the source(s) of anomaly to the level of functional groups/components and practicality is its key feature. **DiVa** takes the data of interest as input and outputs i) a set of time-series data indicating potential anomaly sources(s) as shown in Fig. 3.1 and ii) a list of potential anomaly sources sorted by the length of time window during which **DiVa** determined the data to be anomalous based on the system user/consumer's need.

Besides limited data availability, there are two main technical challenges in developing **DiVa**:

C1: Detailed system parameters are not easy to acquire without the first-party's support. Prior solutions usually require a detailed system model for the design of residual signals, which is proprietary to car-makers. Even with the first-party's support, those

parameters may still not be the same even for the same car model due to customization, adjustment and component replacement. The system must, therefore, learn the normal behavior of each SA system.

C2: Since the mechanisms of identifying anomalous data in prior approaches [8, 18, 47, 55, 89, 99, 143, 156] usually have direct ties to their (detailed) model formulation, limited data availability will also degrade/disable their ability of identifying anomaly sources and there is no easy way to modify their formulation without re-designing the system.

To avoid the need for detailed causality modeling that requires difficult-to-access data for the identification of anomalous data, **DiVa** features *separation* of the identification process from the detection process, rendering the former independent of the causal correlation between data used in the latter. That way, **DiVa** further removes the observability requirement[2] for narrowing down the search space for anomaly sources. This design also facilitates easy upgrade of **DiVa** if engineers or car-makers want to expand **DiVa**'s detection to include more types of data in the future.

Like all other data-driven approaches, **DiVa** also takes time to observe vehicle behavior before outputting its detection result, and hence it is designed to detect early signs of abnormal behavior so as to prevent future accidents/failures that origin from the same vulnerability, instead of providing a real-time defense or fix. **DiVa**'s main purpose is to signal/tell (*e.g.*, periodically or after a trip) which of vehicle components may be anomalous and provide vehicle owners/mechanics hints to locate and fix the anomaly source(s) as efficiently and as quickly as possible.

For the case study of semi-autonomous vehicles (SAVs), since there are only limited data types that can be accessed through the IVN (*e.g.*, CAN bus) or standard OBD-II messages [26, 115], **DiVa** makes best of those data that are commonly accessible through the IVN (i.e., without requiring unavailable or hard-to-access data) (Chapter 3.4). Specifically,

[2]Having sufficient observers or outputs (that can limit the degrees of freedom in system models) to identify internal system state. See Chapter 3.2 for more on this.

DiVa focuses on the detection of two major vehicle maneuvering behaviors:

- *Input-to-response consistency:* how control inputs of the vehicle (*i.e.*, gas pedal, brake, and steering) affect the vehicle dynamics (*i.e.*, speed, acceleration, yaw rate); and
- *Powertrain operation:* how gear level and engine RPM react to the control inputs and how they affect the vehicle dynamics.

This chapter makes the following contributions:

- Development of **DiVa**, a forensic system and a concrete case study of SAVs, that include
  - (Meeting C2) An efficient algorithm for anomaly source identification with formal mathematical formulation (*i.e.*, a Boolean equation system) and performance proof (Chapter 3.4);
  - (Meeting C1) Detection sets (DSs) to capture anomalous vehicular behaviors and their formal formulation for anomaly detection (*i.e.*, the distance function D defined in each DS) with limited data availability and without the requirement of knowing detailed vehicle specifications (Chapters 3.5, and 3.6); and
- Evaluation of **DiVa**'s performance with the real-world data collected from 2 test vehicles and 1 publicly available data set, demonstrating **DiVa**'s i) performance as an anomaly detection system, ii) ability to narrow down the search space for the anomaly source(s), and iii) end-to-end performance as a forensic/diagnostic system (Chapter 3.7).

3.2 Related Work

While the focus of **DiVa** is data-driven anomaly detection and source identification, its related work can be categorized into *Fault Detection and Isolation* (FDI) and general *Anomaly Detection* (AD) in either vehicle systems or generic control systems (*e.g.*, robotic vehicles and industrial control systems). The authors of [46] and [150] conducted comprehensive surveys on prior FDI and AD studies in general control systems, respectively, and the authors of [3] and [145] also conducted comprehensive surveys on security of IVNs,

including the summaries and comparisons of detection scope and performance of different schemes. So, here we only introduce the work that has similar detection scope and does not require any cloud support as **DiVa**.

Fault Detection and Isolation (FDI): FDI [8, 18, 47, 55, 89, 99, 143, 156] focuses on the detection and diagnostics of anomalies caused only by component faults. They usually model the system of interest in detail (*e.g.*, by Bond Graph [72]) and design observers to capture any occurrence of inconsistency and formulate a system of (differential) equations while treating the faults as unknown variables [8, 47, 156]. The faults can then be identified by solving for the unknown variables in the (differential) equation system. They are able to pinpoint the faulty components efficiently, but require detailed knowledge of the target system (*i.e.*, design, architecture, and parameter settings) and detailed system state measurements and control setpoints to function as intended.

Specifically, FDI approaches usually have the following formulation:

$$x[k+1] = A(x[k]) + B(u[k]) + \xi_p[k] + \mu[k] \tag{3.1}$$

$$y[k] = C(x[k]) + \xi_m[k] + \psi[k], \tag{3.2}$$

where x is the ground-truth system state, y is the measurement output, u is the control input/setpoint, μ is the process noise, ψ is the measurement error, and ξ_p and ξ_m are the attack vectors (i.e., that specifies the attack magnitude) corresponding to system process and measurement, respectively. A, B, and C are the transformations that capture the correlation/causality between system state, control input and measurements. The above formulation treats u as a trusted/known data and, therefore, the attack vectors can be obtained by solving the equation system $\mathcal{E}$ of Eqs. (3.1) and (3.2).

If we use incorrect control input as "u" when they can also be manipulated, the resulting ξ_p and ξ_m may be incorrect. However, if we treat them as elements in y, there will be insufficient information in $\mathcal{E}$ to determine uniquely (i.e., there can be multiple solutions

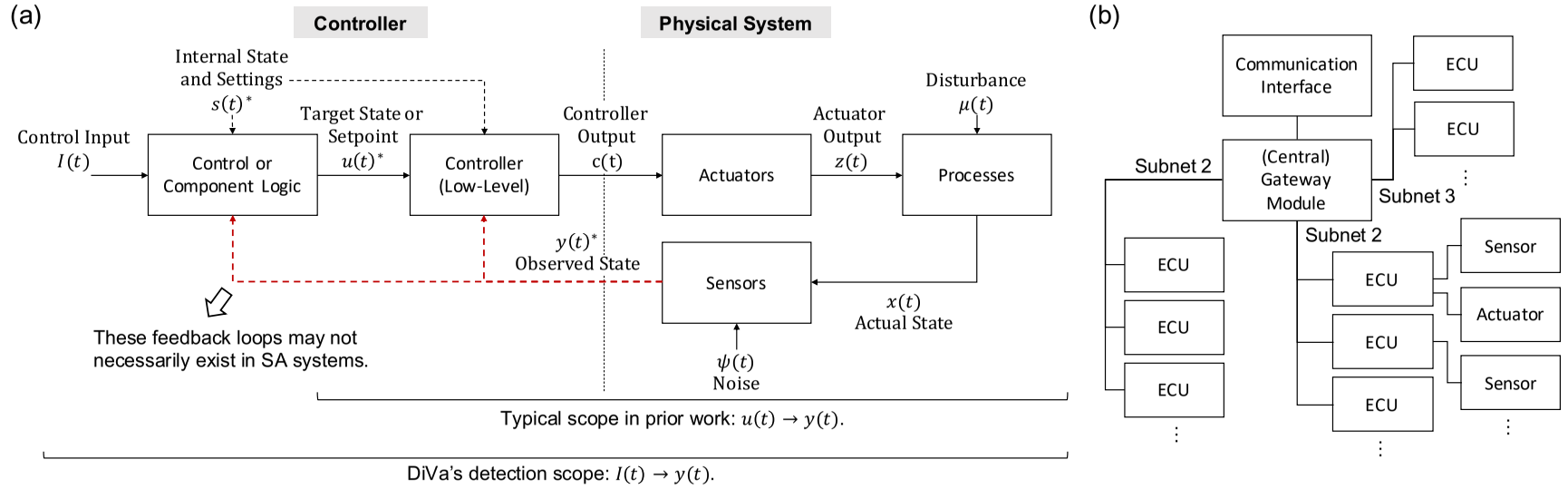


Figure 3.2: (a) A typical control system: the dashed lines indicate a connection may not exist between two blocks and the symbols with "*" indicate the data required by prior work that are partially or entirely missing on the IVNs of commercial SA vehicles. (b) An example IVN.

that can satisfy $\mathcal{E}$). Prior work (e.g., [122]) usually transforms $\mathcal{E}$ further to an optimization problem and solve it numerically and this will make the solving process to stop/stuck at the first solution it finds without identifying other potential anomaly sources.

The main difference between **DiVa** and a typical FDI is that **DiVa** identifies or narrows down the source of an anomaly/fault by solving a Boolean equation system derived from the results of its detection phase. That way, **DiVa** does not rely on the causal correlation to identify the anomalous data as in vehicle FDI approaches, enabling it to correctly identify the anomalous source under an attack (*e.g.*, data manipulation), and nor does it require detailed system layout and measurements to work.

General Anomaly Detection (AD): AD schemes [2, 5, 23, 45, 50, 54, 79, 90, 101, 102, 124, 146, 153, 157] explore the correlation between multiple vehicle data and construct their normal models/behaviors for anomaly detection. There have been proposals to detect anomalies in a single component or functional group (*e.g.*, engine [50, 90, 101, 102], brake [23, 79] and gear system [146]). They are specialized to model the components of interest based on the mechanical design and physical correlation of the components. They can achieve accurate detection, but are difficult to deploy due to their requirement of detailed in-component measurements that are not commonly available on in-vehicle networks.

There have also been system/control/physical invariant approaches proposed to model the control behavior of a cyber-physical system (CPS) for anomaly detection. They usually model the CPS as a typical control system shown in Fig. 3.2a and develop mechanisms to capture the system's control behavior. Choi *et al.* [25] proposed a detection framework by capturing the causality among target state/setpoint input $u(t)$, internal system state $s(t)$, and observed system state $y(t)$ based on a PID controller model, where $u(t)$ and $y(t)$ are designed to point to the same measurement in its design. Similarly, Quinonez *et al.* [104] proposed use of a nonlinear physical invariant for detecting anomalous system behavior, *i.e.*, $u(t) \rightarrow y(t)$. Although the above approaches are applicable to robotic/customized vehicles, they cannot be applied to commercial SA systems, such as commercial vehicles,

due to the lack of data and/or system models they require. Specifically, the data commonly available on IVNs are limited to i) specific control inputs $I(t)$, ii) partial powertrain component state $s(t)$, and iii) dynamics measurements $y(t)$. In other words, most of the data they require, such as (the target and/or measured) wheel/brake torque, are not available on IVNs.

To overcome the absence/lack of efficient models/mappings from $I(t)$ to $y(t)$, the authors of [45] and [54] cross-validated vehicle data in different functional groups to ensure that in-vehicle data are authentic by focusing on the pair-wise correlation between data types. While they aim at anomaly detection in vehicle maneuvering behavior, the authors only considered pair-wise, linear correlations between vehicle data without providing any effective way to identify the sources of anomalies. From a technical point of view, EVAD [54] (*i.e.*, the approach with the best performance among prior data-driven approaches) performs detection based directly on the pair-wise correlation coefficients and power spectrum densities. However, it still requires closely-coupled data that are not commonly available on IVNs (*e.g.*, $\langle$brake torque, brake pedal$\rangle$ and $\langle$wheel torque, acceleration$\rangle$) to achieve effective detection. Dash *et al.* proposed PID-Piper [28] based on long short-term memory (LSTM) machine learning for controller behavior modeling without the requirement of detailed physical modeling. However, as we will show in Chapter 3.7, this approach cannot effectively capture vehicle's behavior for anomaly detection with limited data availability.

In contrast, **DiVa** constructs normal behavior models based on commonly available data on IVNs. It is important to note that data availability is not an easy requirement to meet because IVNs (*e.g.*, CAN bus) are usually designed to meet the real-time requirements of messages using priority-based network access. Adding new data may require a re-design of message priority assignment, which may, in turn, require the modification of multiple ECUs [29, 100]. Also, since ECUs are usually deployed on different in-vehicle (sub)networks of different types (Fig. 3.2b), accessing detailed data from different IVNs may require a fundamental change of IVN architecture which is infeasible/unacceptable for cost reasons.

3.3 Threat Model and Case Study

3.3.1 Threat Model

DiVa is designed to detect anomalies in *system behavior*, not attack patterns, and identify or narrow down their sources from the cyberspace. Specifically, the anomalies considered in **DiVa** can occur under one of the following conditions:

A1. A change of the system's reaction to control inputs owing to a faulty or compromised component;

A2. A sensor failure or a sensor spoofing attack; or

A3. A data manipulation attack from a compromised component.

Data Manipulation (DM) is defined as the act of modifying the value of a message transmitted on the in-system networks (*e.g.*, IVNs) to a (wrong) value that does not represent the actual system state. The main difference between anomalies caused by data manipulation and a component failure is that the anomalous operations of the latter will propagate to other components in the control flow. In contrast, a data manipulation may not have such a causality relationship between the input and the output since data on IVN are not necessarily directly involved in control components (*e.g.*, only for diagnostics or information display). Note that data manipulation is not equivalent to *Data Injection* (DI), *i.e.* the detection target of IDSes [3, 145]. They may lead to similar consequences from a control perspective when the injected messages are also under manipulation attacks, but data injection will generate *additional* messages, which will inevitably change the message transmission pattern.

Since insufficient data availability prevents **DiVa** from accurately capturing all aspects of system state, and also different vehicles are equipped with different types/levels of redundancy, **DiVa** does not rely on the assumption of a fixed, trusted (set of) data. As a common and natural limitation of all data-driven approaches that do not rely on any single trustworthy data, **DiVa** is designed to detect anomalies as long as the adversary cannot

compromise *all data at once* and identify their sources if the majority of its input data are not manipulated coordinately.

3.3.2 Case Study

We use data types (DTs) to indicate a specific data types from the SA entity and avoid confusion with a single data sample. **DiVa**'s case study focuses on the following *vehicular DTs (or VDTs)* that are related to vehicle maneuvers:

- *Control Inputs* (CI) — throttle or gas pedal position (g_a), brake position (b_r), and steering wheel angle (θ_S);
- *Powertrain Components* (PC) — engine RPM (r_m) and gear level (g_e); and
- *Vehicle Dynamics* (DYN) — longitudinal acceleration (a_X), lateral acceleration (a_Y), speed (v) and yaw rate (ω_Z).

DiVa chose these VDTs based on the standard vehicle repair procedure/information (*i.e.*, OBD-II) and the common data availability without significant modification in the vehicle architecture or the model-specific parameters of the vehicle.

The limitation of **DiVa**'s detection described in Chapter 3.3.1 should not diminish **DiVa**'s value and practicality for two reasons. First, as we will later demonstrate in Chapter 3.7, the prior work with the best reported performance still cannot effectively detect a single behavior anomaly with limited data availability. Furthermore, most, if not all, of prior approaches designed for detecting stealthy attacks are not directly applicable to commercial vehicles without the mechanisms/models proposed in **DiVa** to detect naïve anomalies, let alone the detection of stealthy ones. Second, the VDTs covered in **DiVa** are associated with at least three different ECUs (*e.g.*, engine control module, transmission control module, and gateway module) and the recently reported attack only showcases data injection/manipulation through *a single ECU* that has communication interfaces with external entities, *i.e.*, the infotainment system. A successful attack that can evade **DiVa**'s detection/identification requires the attacker to manipulate all (or a great majority) of the

data in **DiVa** by compromising or matching the transmission characteristics of *all* ECUs involved with the target data *simultaneously*. However, using a single ECU to mimic other ECUs' transmissions or generating additional messages to coordinate the attack can easily be caught by the existing IDS/DI approaches [3, 22, 145].

3.4 System Design

3.4.1 Detection Sets

Instead of considering all data as a single correlation group for system behavior modeling, **DiVa** performs anomaly detection on a set of combinations of DTs, called *detection sets* (DSs). DSs are so designed that each of them may capture one or more important system characteristics and the data of each detection set may overlap with those of other DSs for cross-validation of data integrity.

DS Index	Type	CI			PC		DYN			
		g_a	b_r	θ_S	r_m	g_e	a_X	a_Y	ω_Z	v
$*$ 1	CI×DYN	✓	✓				✓			✓
2		✓					✓			✓
3			✓				✓			✓
$*$ 4	PC×DYN				✓	✓				✓
5					✓					✓
6						✓				✓
7	CI×PC	✓	✓		✓					
8		✓	✓			✓				
9	All	✓				✓				✓
$*$ 10	CI×DYN			✓				✓		✓
$*$ 11				✓					✓	✓
12	DYN							✓	✓	✓

Table 3.1: Detection sets of **DiVa**, where $*$ indicates an always-on detection set.

Table 3.1 summarizes 9 VDTs $(g_a, b_r, \ldots, v)$ and 12 DSs **DiVa** considers in its case study. Specifically, DS_1–DS_3 capture the input-to-response consistency related to the vehicle's acceleration in longitudinal direction while DS_{10}–DS_{12} capture the lateral maneuver characteristics of steering control. DS_4–DS_6 capture the relation between the output of

powertrain components and their resulting vehicle dynamics. DS_7 and DS_8 capture how the control inputs interact with powertrain components. Finally, DS_9 models the automatic transmission's behavior.

These 12 DSs capture the most fundamental vehicle operations and, therefore, are directly applicable to the vehicles commonly seen on the roads (*e.g.*, sedans and SUVs). Furthermore, if a first-party carmaker wants to integrate **DiVa** into its vehicles, more detailed DSs can be added to the existing DSs for more fine-grained identification. In case more fine-grained data become available, we provide a performance proof in Chapter 3.4.4.3.

3.4.2 Overview of System Workflow

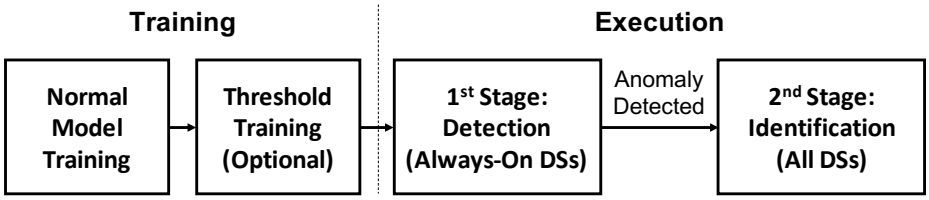


Figure 3.3: **DiVa**'s workflow.

DiVa's basic workflow consists of *training* and *execution* phases as shown in Fig. 3.3. In the training phase, **DiVa** monitors/collects the data in the SA system and constructs the system's normal behavior model for each detection set. After constructing the normal behavior models, **DiVa** provides an option for system developers to automatically derive the thresholds to be used during run-time. During the execution phase, **DiVa** performs anomaly detection *continuously*. It first detects anomalies based on some selected DSs. To filter out transient false positives, **DiVa** reports an anomaly only after α consecutive positive detections.[3] If an anomaly is detected, **DiVa** attempts to identify the anomaly source/cause by combining the detection results from all the DSs.

This two-stage mechanism is designed based on the fact that not every detection set holds the same level of causal relation (or correlation) between their data and some of the

[3] α is a design parameter that can be set to the developer's/user's preference or obtained during the optional parameter training.

detection sets are available only when a certain event occurs, *e.g.*, DS_2 can only work when the gas pedal is pressed. The detection sets selected for reporting an anomaly (*i.e.*, the 1st stage) are DS_1, DS_4, DS_{10}, and DS_{11}, which we call *always-on* detection sets. By performing detection only on always-on detection sets first, **DiVa**'s overall false positive rate will not be influenced by the false positives from other DSs with weaker causal correlation. In the case study, we choose always-on DSs because they:

- collectively cover all the VDTs considered in the case study;
- all have a common VDT (*i.e.*, speed "v"); and
- have deterministic causal correlation/models between their VDTs.

Other than reducing false positives, the essence of selecting these always-on DSs is to prevent a coordinated or colluded attack from going undetected. If an attacker wants to mount certain attacks that can fool an always-on DS, s/he must also manipulate/match speed. In the meantime, other always-on DSs will detect the anomaly as long as the attack does not manipulate *all* of the data coordinately, which is not in the scope of this thesis, and is impossible for any data-oriented system to detect without assuming certain trusted data.

After an anomaly is detected in the 1st stage, **DiVa** performs anomaly detection on the remaining DSs and executes its source identification algorithm based on the detection results from all the DSs to narrow down the search space of anomalous data source.

3.4.3 Anomaly Detection in Detection Sets

We now introduce two mechanisms that **DiVa** utilizes to perform anomaly detection.

3.4.3.1 Detection by Distance to Normal Model

This approach is designed to identify system behaviors that significantly deviate from normal operation. Assume a DS is governed by a model $\mathcal{M}_{tr}$ in the form of

$$x'(t) = \mathcal{F}(x_1(t), x_2(t), \ldots, x_n(t)), \tag{3.3}$$

where $x_1(t), x_2(t), \ldots, x_n(t)$ are data in the DS and $x'(t)$ is another data in the same DS or a derived characteristic.

DiVa utilizes $\mathcal{M}_{tr}$ and the testing data $x_{te,1}(t)$, $x_{te,2}(t)$, ..., $x_{te,n}(t)$ to obtain an estimation of $x'_{te}(t)$, denoted as $\widehat{x'_{te}}(t)$. **DiVa** then computes $d = D(x_{te}, \widehat{x'_{te}})$ to $\mathcal{M}_{tr}$, where $D(\cdot)$ is the function that compares $\widehat{x'_{te}}$ and x'_{te} and outputs a scalar d, indicating how much x'_{te} deviates from $\mathcal{M}_{tr}$. Finally, **DiVa** reports an anomaly if d is greater than a threshold Γ. $D(\cdot)$ can be expressed with the notion of tuple using multiple distance definitions and threshold values. In a given $D(\cdot) = \langle D_1(\cdot), D_2(\cdot), \ldots \rangle$, **DiVa** will report an anomaly if any distance element exceeds its corresponding threshold.

3.4.3.2 Detection by Model Change

This detection is designed to identify a minor change of behaviors over a long period of time that is undetectable by the previous approach. This detection is performed periodically to ensure that **DiVa** can capture minor changes over a long period of time. Like the previous example, suppose **DiVa** obtains the model $\mathcal{M}_{tr}$ with model parameters $\{p_{tr,1}, p_{tr,2}, \ldots, p_{tr,k}\}$ after the training phase. **DiVa** takes the testing data $\{x_{te,1}(t), x_{te,2}(t), \ldots, x_{te,n}(t)\}$ as inputs, uses the same model training procedure as in the training phase, and derives a testing model $\mathcal{M}_{te}$ with model parameters $\{p_{te,1}, \ldots, p_{te,k}\}$. **DiVa** compares the parameters of training and testing models one-by-one and reports an anomaly detection if any $|p_{tr,k} - p_{te,k}| > \Gamma_{M,k}$.

3.4.4 Identification of Anomaly Source(s)

Let us first introduce some notations before presenting the identification algorithm. Let DS_z be a DS with index z, $DS_z \rightarrow \checkmark$ ($DS_z \rightarrow \boldsymbol{\times}$) represents the case when DS_z succeeds (fails) in matching the normal behavior, and $ds_z(t)$ be the time-series data of DS_z.

3.4.4.1 Problem Formulation

Our goal is to classify DTs (*i.e.*, $x_1, \ldots, x_9$) as a potential anomalous set (P) or a correct set (Q). $x_i \in P$ indicates that $\bar{x}_i$ (*i.e.*, an indicator of x_i) can be either anomalous (0) or correct (1), while $x_i \in Q$ indicates that $\bar{x}_i$ can only be equal to 1. The problem of identifying an anomaly source can be formulated as that of finding P and Q that satisfy the system of Boolean equations constructed based on the detection results on DSs:

$$\begin{cases} \bigwedge\limits_{x_i \in DS_j} (\bar{x}_i) = 1, \text{ if } DS_j \rightarrow \checkmark & (3.4) \\ \bigwedge\limits_{x_i \in DS_j} (\bar{x}_i) = 0, \text{ if } DS_j \rightarrow \boldsymbol{\times}, & (3.5) \end{cases}$$

where $\bigwedge$ denotes *AND* operation. Since only the elements in P can produce a 0 detection and Eq. (3.4) can always be satisfied, this problem can be simplified further as that of identifying all possible $x_i \in P$ such that

$$\bigvee_{x_i \in DS_j} (\neg \bar{x}_i) = 1, \text{ if } DS_j \rightarrow \boldsymbol{\times}, \tag{3.6}$$

where $\bigvee$ denotes *OR* operation. That is, the goal is to identify at least one anomalous DT in a DS upon detection of an anomaly. While this formulation may seem similar to the hitting set problem [30, 31, 97], there exist two main differences that make the identification problem unique. First, the hitting set problem does not consider the condition in which there can be incorrect detection (*i.e.*, false positives or negatives). Second, our goal is to find the DT that is most likely to be anomalous during the period of interest, instead of

finding a minimum number of anomalous DTs from a single detection.

3.4.4.2 Identification Algorithm (IA)

We have developed an efficient algorithm to solve Eq. (3.6) with computation complexity of $O(NK)$, where N (K) is the number of DTs (DSs).

The first step of anomaly source identification is to combine the detection results in DSs by creating a table as shown in Table 3.2 and marking each cell according to their results. **DiVa** labels all the DTs in DS_z to be 1 if $DS_z \rightarrow \checkmark$, and 0 otherwise. The value 1 means that the DT matches the normal behavior while 0 means the DT's integrity to be determined. Then, **DiVa** performs a column-wise OR operation on the result table as shown in the "All" row of Table 3.2 and outputs all the DTs with value 0 as potential anomaly source(s).

Ideally, the detection results should correctly capture the anomaly in each DS. However, it is possible that some DSs have incorrect detections. Therefore, **DiVa** performs a sanity check before generating the final detection result. Erring on the safer side of detection, **DiVa** is biased to determine that a DT is a potential anomaly source when a *conflicting detection* (CD) occurs. A CD is defined as the condition that a DS_z does not match its normal behavior, but all other DSs that have overlapping data with DS_z pass the detection. In case of a CD, **DiVa** will set all the data in DS_z to be anomalous.

DiVa performs the above procedure and marks the potential anomaly source(s) whenever an anomaly is detected. Users will be able to select the time period of their interest and obtain a summary of that period. For example, if a user is interested in his last trip, **DiVa** will create a chart as shown in Fig. 3.1 and rank the vehicle data according to their likelihood of being the anomaly source(s) based on the time **DiVa** determined them to be anomalous (called *anomalous time*).

DS	Result	g_a	b_r	θ_S	r_m	g_e	a_X	a_Y	ω_Z	v
1	✓	1	1	-	-	-	1	-	-	1
4	✗	-	-	-	0	0	-	-	-	0
10	✓	-	-	1	-	-	-	1	-	1
11	✓	-	-	1	-	-	-	-	1	1
2	✓	1	-	-	-	-	1	-	-	1
3	✓	-	1	-	-	-	1	-	-	1
5	✓	-	-	-	1	-	-	-	-	1
6	✗	-	-	-	-	0	-	-	-	0
7	✓	1	1	-	1	-	-	-	-	-
8	✗	0	0	-	-	0	-	-	-	-
9	✗	0	-	-	-	0	-	-	-	0
12	✓	-	-	-	-	-	-	1	1	1
All	✗	1	1	1	1	0	1	1	1	1

Table 3.2: Example of combining detection results while assuming only gear data (g_e) is anomalous.

3.4.4.3 Detection Correctness and Performance Guarantee

Before discussing the properties of the identification algorithm (IA), we first introduce some terminologies. *Correct detection* indicates the condition that the consistency check (*i.e.*, whether or not the data passes the anomaly detection) will fail *iff* at least one of the DTs in a DS is anomalous. *Weak detection* indicates the condition that the consistency check will pass if all the DTs in a DS are correct (*i.e.*, without manipulation) and the consistency check will fail if not all the data in a DS are manipulated. A representative condition of weak detection is the adversary launching a sophisticated attack that targets a certain set of DSs to evade **DiVa**'s detection. The correctness and effectiveness of IA can be stated as Properties 3.4.1–3.4.2 and 3.4.3–3.4.5, respectively. Note that the properties discussed here are the general characteristics of IA and they always hold irrespective of DS design under consideration (even when including more than the original ones) as long as there exists a set of always-on DSs such that all DSs in this set (i) collectively cover all the DTs and (ii) share a common DT.

Property 3.4.1. *(Detection Guarantee) If not all DTs are manipulated, IA is guaranteed to detect anomalies under weak detection.*

Proof. Since not all DTs are manipulated, there must exist an always-on DS that contains at least one correct DT and one manipulated DT because of the DS requirement described earlier. According to the definition of weak detection, the DS contains at least one correct DT and one manipulated DT will fail the consistency check. Therefore, the anomaly will be detected. □

Property 3.4.2. *(Correctness) The result obtained from IA (i.e., P and Q) always satisfy Eq. (3.6) under weak detection.*

Proof. Assume there is a solution of IA that does not satisfy Eq. (3.6). That is, $\exists\ k$ s.t. $\bigwedge_{x_i \in DS_k}(\bar{x}_i) = 1$ always holds and $DS_k \rightarrow$ ✗. This indicates all $x_i \in DS_k$ are in Q, which violates the condition of sanity check, and hence this solution cannot be a result from IA. Therefore, the assumption must be false. □

Property 3.4.3. *(Suspect with Reason) Under weak detection, there must exist some $DS_j \rightarrow$ ✗ and $x_i \in DS_j$ for all x_i determined to be in P.*

Proof. Assume $\exists\ x_i \in P$ s.t. $\forall DS_j \ni x_i$, $DS_j \rightarrow$ ✓. Since $DS_j \rightarrow$ ✓ for all $DS_j \ni x_i$, $\bar{x}_i$ must be 1 before the sanity check. Also, the sanity check will only change $\bar{x}_i$ to 0 if x_i in one of the DSs that fail the consistency check. Therefore, x_i must be in Q. This result contradicts the assumption, and hence the assumption must be false. □

Property 3.4.4. *(Guarantee of Correct Detection) IA is guaranteed to include all the anomalous data in P under correct detection.*

Proof. Assume $\exists\ x_i$ s.t. $x_i \in Q$ and x_i is anomalous. Since IA will only classify a DT in Q if $\exists\ DS_j$ s.t. $DS_j \rightarrow$ ✓ and $x_i \in DS_j$, this contradicts the assumption of correct detection that the consistency check will always fail if there is at least one anomalous data in the DS. Therefore, the assumption must be false. □

Property 3.4.5. *(Guarantee of Weak Detection) Under a weak detection, IA guarantees that all the anomalous data will be included in P as long as they are not in a DS that is under a coordinated attack.*

Proof. Assume $\exists\ x_i$ s.t. $x_i \in Q$ and x_i is anomalous. Since IA will only classify a DT in Q if $\exists\ DS_j$ s.t. $DS_j \rightarrow \checkmark$ and $x_i \in DS_j$, this contradicts the assumption of weak detection that the consistency check will always fail if there is at least one anomalous data in the DS when not under a coordinated attack. Therefore, the assumption must be false. □

Property 3.4.3 guarantees that IA will not include unnecessary data in P while Properties 3.4.4 and 3.4.5 make sure that IA will include all the suspected anomalous data in P. In what follows, we explore the DSs proposed in **DiVa** and describe how they can be used to model the normal vehicle behavior for anomaly detection.

Next, we detail the DSs used in **DiVa**'s case study.

3.5 Input-to-Response Detection Sets

The models in this class of DSs describe the causal relations between the control inputs and the vehicle dynamics. Specifically, the model $\mathcal{M}$ has the form of:

$$x_p(t) = \mathcal{F}(x_1(t), \ldots, x_{p-1}(t), x_{p+1}(t), \ldots, x_n(t)), \tag{3.7}$$

where $\{x_1, \ldots, x_n\}$ is the set of VDTs in the DS and x_p is a VDT of DYN.

3.5.1 DS_1 $\{g_a, b_r, a_X, v\}$

Model: DS_1 explores the causal relation between the control inputs and the vehicle dynamics in longitudinal direction. Specifically, DS_1 describes how longitudinal acceleration will change if the gas pedal or brake is pressed. According to Newton's second law and the general vehicle's physical model [82], the vehicle's acceleration can be expressed as the

effect of the engine's drive force (F_E), braking force (F_B), resistance force (F_R), including aerodynamic drag and wheels' rolling drag, and other unobservable contexts (F_O):

$$m\vec{a}_X = \vec{F}_E + \vec{F}_B + \vec{F}_R + \vec{F}_O. \tag{3.8}$$

F_E (F_B) is determined by the vehicle state and the amount of gas (brake) pedal pressure applied while F_R is determined solely by the vehicle state. Therefore, the model of DS_1 ($\mathcal{M}_{DS_1}$) can be described as:

$$\begin{aligned}\hat{a}_X(t) = B_0(v(t)) + B_g(v(t), g_a(t))\ K_g(g_a(t)) \\ + B_b(v(t))\ K_b(b_r(t)),\end{aligned} \tag{3.9}$$

where K_g and K_b are the functions that reflect how the pedal positions influence the acceleration; B_0, B_g, and B_b represent the scaling effects which depend on the state of the vehicle. Note that the rationale behind the design of $B_g(\cdot)$ is related to capturing the gear transition behavior (Chapter 3.6.6).

We then approximate the functions in Eq. (3.9) with their Taylor expansions:

$$\begin{aligned}\hat{a}_X(t) = B_0(v(t)) + \sum_{i=0}^{N_{g,B}} B_{g,i}\,(v(t))\,g_a^i \times \sum_{p=0}^{N_{g,K}} k_{g,p}\ g_a^p \\ + \sum_{j=0}^{N_{b,B}} B_{b,j}\,(v(t))\,b_r^j \times \sum_{q=0}^{N_{b,K}} k_{b,q}\ b_r^q\end{aligned} \tag{3.10}$$

where the N's are the orders of the approximation. Eq.(3.10) can be further simplified as:

$$\begin{aligned}\hat{a}_X(t) = C_0(v(t)) + \sum_{i=1}^{N_g} C_{g,i}(v(t))g_a^i + \sum_{j=1}^{N_b} C_{b,j}(v(t))b_r^j, \\ \text{where } N_g = N_{g,B} + N_{g,K} \text{ and } N_b = N_{b,B} + N_{b,K}.\end{aligned} \tag{3.11}$$

The training of DS_1 can then be performed by identifying the scaling effect (*i.e.*, the C's)

with regard to v based on machine learning techniques, such as ridge regression.

We introduce the process of training C_0, $\{C_{g,i}\}$, and $\{C_{b,j}\}$. **DiVa** utilizes a similar training process as **CADD**, but without road inclination calibration, to identify the parameters based on their values at different speeds. Specifically, it partitions the data into small segments with a predefined time window (10s in our implementation) and performs ridge regression [12] to estimate C_0, $\{C_{g,i}\}$, and $\{C_{b,j}\}$ in each segment:

$$\hat{C}_k = \text{argmin}_{C_k}(a_{X,k} - C_k^T \omega_k)^2 + \lambda |\omega_k|^2, \tag{3.12}$$

where k is the segment index, $C_k = [C_{(0,k)}, C_{(g,1,k)}, \ldots, C_{(X,g,N_g,k)}, C_{(b,1,k)}, \ldots, C_{(b,N_b,k)}]^T$, $\omega_k = [1, g_a(t_{k,1}), \ldots, g_a(t_{k,N_T}), b_r(t_{k,1}), \ldots, b_r(t_{k,N_T})]^T$, N_T is the total number of data in each segment, and λ is the regularization term.

The final step is to find a mapping between the speed and the parameter values. **DiVa** utilizes the average speed as the speed in the target segment. That way, we can plot the values of parameters and speed on a 2D space, and identify the mapping by performing regression. Fig. 3.4 shows an example of using a third-order polynomial to approximate the mapping between the parameters and the vehicle speed. In this example, we can observe that the data points form clear patterns, indicating our model's ability to correctly capture the correlation between the data considered in DS_1.

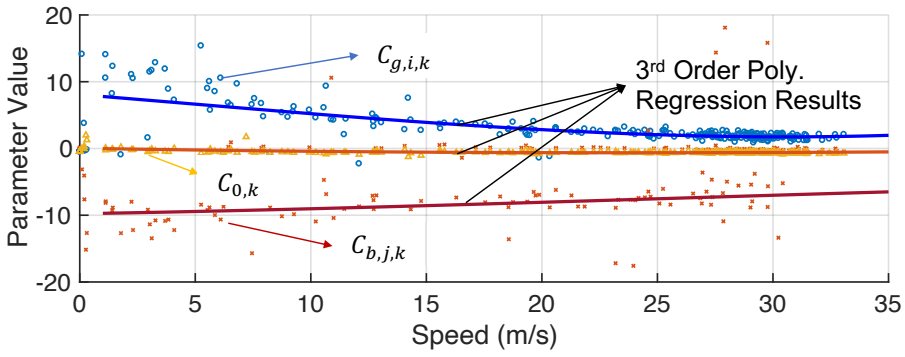


Figure 3.4: Example of estimating parameters of DS_1 model. The data in this example are extracted from Comma AI's open dataset [27].

The D function of DS_1 is defined as the absolute difference between the predicted and the actual acceleration:

$$D_{DS_1} := |\hat{a}_X(t) - a_X(t)|. \tag{3.13}$$

3.5.2 DS_2 $\{g_a, a_X, v\}$ and DS_3 $\{b_r, a_X, v\}$

The models of DS_2 and DS_3, denoted as $\mathcal{M}_{DS_2}$ and $\mathcal{M}_{DS_3}$, respectively, are simplified versions of $\mathcal{M}_{DS_1}$. Instead of considering both gas and brake pedals at the same time, $\mathcal{M}_{DS_2}$ ($\mathcal{M}_{DS_3}$) focuses on the time when only the gas (brake) pedal is pressed. Since the operations of training and detection are basically the same as in DS_1, we omit their details.

3.5.3 DS_{10} $\{\theta_S, a_Y, v\}$ and DS_{11} $\{\theta_S, \omega_Z, v\}$

DiVa exploits the characteristics of circular motion to model the lateral motion of the vehicle. The yaw rate (ω_Z) and the steering wheel angle (θ_S) of the vehicle follows the following equation [77]:

$$\omega_Z = v \times \sin(\theta_S/k)/l, \tag{3.14}$$

where k is the steering ratio and l is the vehicle length. Specifically, when making turns, the lateral acceleration (a_Y) induced by the centripetal force is proportional to the square of the yaw rate $\times$ the turning radius (r): $a_Y \propto \omega_Z^2 r$. Since r is determined by θ_S, **DiVa** constructs $\mathcal{M}_{DS_{10}}$ and $\mathcal{M}_{DS_{11}}$ as:

$$\begin{aligned} &\mathcal{M}_{DS_{10}} : \hat{a}_Y(t) = b_{Y,1}(v(t))\ b_{Y,2}(\theta_S(t)) \text{ and} \\ &\mathcal{M}_{DS_{11}} : \hat{\omega}_Z(t) = b_{Z,1}(v(t))\ b_{Z,2}(\theta_S(t)). \end{aligned} \tag{3.15}$$

Similarly to $\mathcal{M}_{DS_1}$, $D_{DS_{10}}$ and $D_{DS_{11}}$ are defined as:

$$D_{DS_{10}} := |\hat{a}_Y(t) - a_Y(t)|, \text{ and} \tag{3.16}$$

$$D_{DS_{11}} := |\hat{\omega}_Z(t) - \omega_Z(t)|. \tag{3.17}$$

3.5.4 DS_{12} $\{a_Y, \omega_Z, v\}$

Unlike the other DSs in this category, DS_{12} is designed to check the consistency between the vehicle dynamics in lateral direction. Utilizing the circular motion that the centripetal acceleration is proportional to the product of angular speed and the velocity ($a_Y \propto \omega_Z v$), **DiVa** constructs $\mathcal{M}_{DS_{12}}$ as:

$$\mathcal{M}_{DS_{12}} : \hat{a}_Y(t) = s \times \omega_Z(t) \times v(t), \tag{3.18}$$

where s is the model parameter that is obtained by the ridge regression during the training phase. Fig. 3.5 shows an example of the linear correlation between the VDTs in DS_{12}.

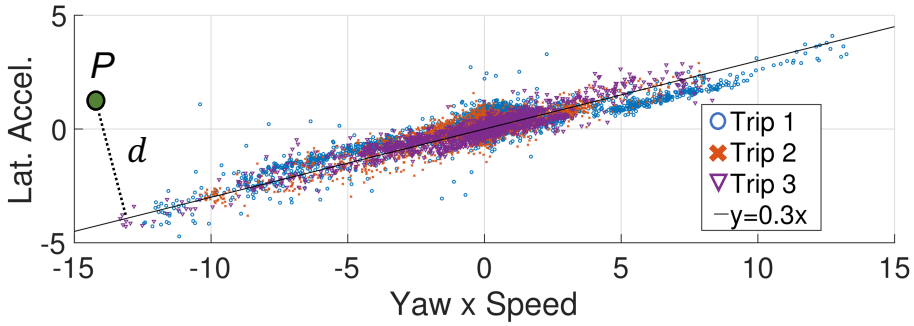


Figure 3.5: This figure shows an example of *Sedan*'s linear correlation between a_Y and $\omega_Z \times v$.

$D_{DS_{12}}$ is defined as the distance between a given data $P = (\xi(t), \eta(t)) = (\omega_Z(t) \times v(t), a_Y(t))$ and the curve $y = sx$:

$$D_{DS_{12}} := d = |\eta(t) - s\xi(t)| / \sqrt{1+s^2}. \tag{3.19}$$

3.5.5 Correlation and Energy Differences

We have thus far discussed DSs with a strong (causal) correlation. While all of their models have the same form as described in Eq. (3.7), a direct comparison of the predicted value ($\hat{x}'(t)$) and the recorded value ($x'(t)$) may sometimes result in a high false positive rate due to transient noises. To account for this, **DiVa** performs additional detection if

$D(\hat{x}'(t), x'(t)) > \Gamma$ within an observation window ($\mathcal{T}_W$=10s). **DiVa** computes i) the correlation coefficient, and ii) the energy difference between $\hat{x}'(t)$ and $x'(t)$, $t \in \mathcal{T}_W$. While the former measurement checks whether the features of $\hat{x}'(t)$ and $x'(t)$ match each other, the latter measurement detects whether there is an extra scaling or shifting between the two data. The energy difference is defined as:

$$E_D = \int_{\mathcal{T}_W} |x'(t)^2 - \hat{x}'(t)^2| dt. \tag{3.20}$$

Similarly, if one of the measurements exceeds the detection threshold, **DiVa** will report the detection of an anomaly.

3.6 Powertrain Operation Detection Sets

For powertrain control operations, **DiVa** constructs models to extract vehicle behavior characteristics (VBC), such as gear-switch timing and ratio of vehicle speed to engine RPM. Specifically, **DiVa** derives the VBC according to the VDTs in the detection sets:

$$\gamma = \mathcal{F}(x_1(t), x_2(t), \ldots, x_n(t)), \tag{3.21}$$

where γ is the target vehicle characteristics.

3.6.1 DS_4 $\{r_m, g_e, v\}$

3.6.1.1 Model:

$\mathcal{M}_{DS_4}$ captures the speed-to-rpm ratio in each gear level. To construct $\mathcal{M}_{DS_4}$, **DiVa** exploits the fact that i) there is a one-to-one mapping between gear levels and the equivalent gear ratio of the gear-train system and ii) the wheel speed is proportional to the product of engine RPM and gear ratio [124]. That is, the vehicle speed (v) will be proportional to the engine RPM (r_m) given a fixed gear level (g_e). Specifically, there exist step-shaped patterns

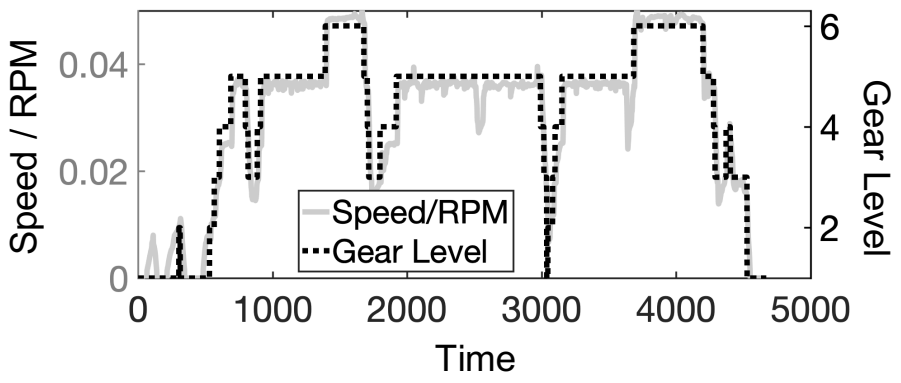


Figure 3.6: $v(t)/r_m(t)$ matches $g_e(t)$ from a real-world trace.

in the observed value of $gr = v/r_m$ that indicate the gear level (Fig. 3.6). **DiVa** captures the average of gr in each gear level to construct $\mathcal{M}_{DS_4}$:

$$\rho(g_e(t)) = v(t)/r_m(t), \tag{3.22}$$

where $\rho(k) = gr_k$ is the mapping between gear level k and speed-to-rpm ratio gr_k.

3.6.1.2 Detection:

DiVa reports detection of an anomaly when speed-to-rpm ratio of testing data does not match the normal model. D_{DS_4} is defined to be the absolute difference between the recorded gr and the estimated gear ratio $\widehat{gr}$ using $\rho(\cdot)$:

$$D_{DS_4} := |gr(t) - \rho(g_e(t))|. \tag{3.23}$$

Similarly, **DiVa** continuously monitors gr_k of the testing data (*i.e.*, $gr_{k,test}$) and report the detection of an anomaly by model change if $gr_{k,test}$ deviates from the normal model.

3.6.2 DS_5 $\{r_m, v\}$

There is no apparent correlation between these two VDTs when we only consider engine RPM and speed. Sometimes they have a high correlation coefficient, while they do not at some other times, depending on whether or not the gear is in transition. To construct $\mathcal{M}_{DS_5}$, **DiVa** computes $gr(t) = v(t)/r_m(t)$ and then determines the gear level at any given time by identifying the closest value of gr_k obtained from DS_4. After determining the gear level, **DiVa** considers only those time periods $\mathcal{T}_s$ when $gr(t)$ stays stable (Fig. 3.7) by filtering out the time when the gear is in transition. $\mathcal{M}_{DS_5}$ is designed to be the average correlation coefficient between speed and RPM in the stable period as well as the range of speed and RPM value in each gear level:

$$\begin{aligned}
\gamma_{v,rpm} &= corr(r_m, v),\ t \in \mathcal{T}_s, \\
\varepsilon_{v,k} &= [\min(v_k(t)), \max(v_k(t))],\ t \in \mathcal{T}_s, \\
\varepsilon_{r,k} &= [\min(r_{m,k}(t)), \max(r_{m,k}(t))],\ t \in \mathcal{T}_s,
\end{aligned} \tag{3.24}$$

where *corr* is the function that computes the correlation coefficient between the input data and subscript k indicates that the time when the gear level is determined to be k.

D_{DS_5} is defined to be the difference of correlation coefficients of speed and engine RPM between the training (γ_{tr}) and testing (γ_{te}) phases and how much the recorded values of v and r_m differ from $\varepsilon_{v,k}$ and $\varepsilon_{r,k}$, respectively:

$$D_{DS_5} := \langle |\gamma_{tr} - \gamma_{te}|,\ v(t) \ominus \varepsilon_{v,k}, r_m(t) \ominus \varepsilon_{r,k} \rangle, \tag{3.25}$$

where $\varepsilon_{X,k} = [\varepsilon_{X,k,min}, \varepsilon_{X,k,max}]$ and $X(t) \ominus \varepsilon_{X,k} =$
$\max(|\max(0, X_{k,max} - \varepsilon_{X,k,max})|, |\min(0, X_{k,min} - \varepsilon_{X,k,min})|)$.

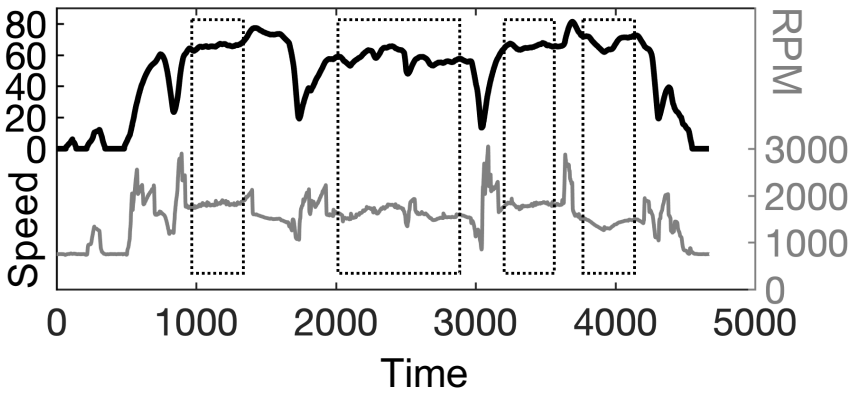


Figure 3.7: Engine RPM (thin curve) and speed (thick curve) show similar patterns when the gear is not in transition (marked by the black boxes).

3.6.3 DS_6 $\{g_e, v\}$

Unlike the condition in DS_5, gear and speed show a strong correlation when the gear is in transition, but not when the gear remains in the same level. $\mathcal{M}_{DS_6}$ has the form of:

$$\gamma_{gear,v} = corr(g_e, v),\ t \notin \mathcal{T}_s. \tag{3.26}$$

D_{DS_6} is defined to be the difference of correlation coefficients of speed and rpm between the training ($\gamma_{tr,gear,v}$) and testing ($\gamma_{te,gear,v}$) phases:

$D_{DS_6} := |\gamma_{tr,gear,v} - \gamma_{te,gear,v}|$.

3.6.4 DS_7 $\{g_a, b_r, r_m\}$

DS_7 is one of the DSs with VDTs that do not have any strong correlation with each other when only the included VDTs are considered. So, **DiVa** does not construct any specific model for the vehicle and performs generic anomaly detection that is applicable to vehicles. **DiVa** monitors the change of engine RPM when three events occur:

- $\mathcal{T}_{gs}$ — the time between the gas pedal is just pressed and the percentage of gas pedal position starts to decrease;

- $\mathcal{T}_{ge}$ — the time between the gas pedal position starts to decrease and the gas pedal is completely released; and
- $\mathcal{T}_b$ — the time when the brake pedal is pressed.

The choice of events is based on the fact that engine RPM will be affected by the change of control input before transmission and, therefore, the *transient* change of engine RPM when these events occur is less likely to be the result of change of transmission. That is, the RPM will increase when the first event occurs and the RPM will decrease when the latter two events occur. D_{DS_7} is defined as the change of RPM during $\mathcal{T}_{gs,p}$, $\mathcal{T}_{ge,q}$, and $\mathcal{T}_{b,r}$ in the detection window, where p, q, and r the event indices:

$$D_{DS_7} := \langle \Delta r_m(\mathcal{T}_{gs,1}), \ldots, \Delta r_m(\mathcal{T}_{ge,1}), \ldots, \Delta r_m(\mathcal{T}_{b,1}), \ldots \rangle. \tag{3.27}$$

In normal condition, $\Delta r_m(\mathcal{T}_{gs,p})$ should be greater than 0; $\Delta r_m(\mathcal{T}_{ge,q})$ and $\Delta r_m(\mathcal{T}_{b,r})$ should be less than 0. **DiVa** will determine that an anomaly has occurred when any of the above relations does not hold.

3.6.5 DS_8 $\{g_a, b_r, g_e\}$

Similarly to DS_7, DS_8 utilizes generic vehicle characteristics that the gear level will only increase when there is a positive acceleration, and decrease when there is a negative acceleration to perform anomaly detection. While the positive (negative) acceleration is the result of applying gas (brake) pedal in most of the scenarios, **DiVa** monitors whether gas/brake pedal is pressed during a gear change.

Specifically, **DiVa** computes the difference of the durations when the gas and brake pedals are pressed and the gear is changing. Therefore, D_{DS_8} is given as:

$$D_{DS_8} := \langle |\mathcal{T}_{g,u,1}| - |\mathcal{T}_{b,u,1}|, \ldots, |\mathcal{T}_{g,d,1}| - |\mathcal{T}_{b,d,1}|, \ldots \rangle,$$

where the subscript g (b) indicates that the gas (brake) pedal is pressed and subscript u (d)

indicates the occurrence of up-shift (down-shift) events. **DiVa** reports the occurrence of an anomaly when $|\mathcal{T}_{g,u,p}| - |\mathcal{T}_{b,u,p}| < 0$ or $|\mathcal{T}_{g,d,r}| - |\mathcal{T}_{b,d,r}| > 0$.

3.6.6 DS_9 $\{g_a, g_e, v\}$

3.6.6.1 Model:

$\mathcal{M}_{DS_9}$ captures the gear shifting behavior (GSB) of the vehicle. The GSB of a vehicle with automatic transmission can be described by a set of curves that mark the threshold of throttle position and vehicle speed when a gear shift occurs [146]. Since the throttle is controlled by the gas pedal, **DiVa** utilizes gas pedal position to capture the GSB. Fig. 3.8 is an example of GSB graph, where the data points are the recorded gear shift events and the dotted curves are the thresholds that the vehicle makes a gear shift. $\mathcal{M}_{DS_9}$ is then designed as:

$$\hat{g}_a(t) = GS(v(t), g_e(t)),\ t \in \mathcal{T}_g, \tag{3.28}$$

where $GS(\cdot)$ is the set of equations that describes GSB threshold curves and $\mathcal{T}_g$ is the time when the vehicle makes gear shifts. D_{DS_9} is then defined to be the minimum distance between the recorded gear shift point P to the corresponding GSB curves. Training $\mathcal{M}_{DS_9}$ is equivalent to obtaining $GS(\cdot)$ for each gear level k given the training data in each gear level $\{(v, g_a)_k\}$ by performing piece-wise regression [12].

3.6.6.2 Training:

Suppose there is a set of training data $\{(v, g_a)_k\}$, a record of speed and gas pedal position when the vehicle makes a certain gear shift (e.g., gear level 1 to 2). Since GSB of the vehicles is a car maker's proprietary design that does not have any standard, it is challenging to model the GSB with limited training data. Furthermore, since the GSB curves may have segments that are parallel to either of the axes of v or g_a, a naive piece-wise regression will not work in this case.

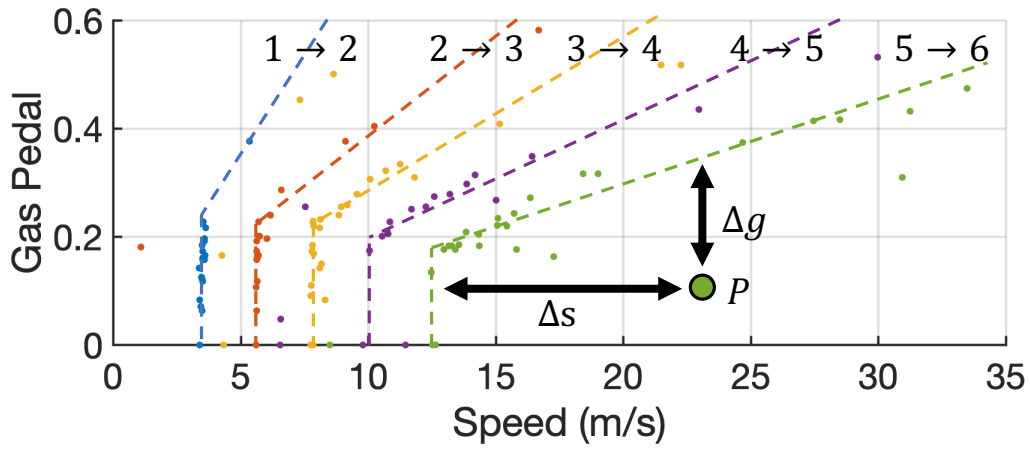


Figure 3.8: Upshift curve of *Sedan*'s gear transition.

To meet these challenges, **DiVa** needs to have an effective way to 1) divide the training data into segments, 2) perform some regression algorithms, and then 3) combine the results from the segments to form a complete GSB curve.

Divide data into segments: The first step of training $\mathcal{M}_{DS_9}$ is to normalize v and g_a to the range of $[0, 1]$ to avoid any computational bias towards one of the data. **DiVa** then performs principal component analysis (PCA) to identify the first principal component ($\vec{PC}_1$) in the speed–gas plane where the data varies the most. **DiVa** then projects all the training data on $\vec{PC}_1$ and divides them into a predefined number of segments.

Perform regression on the ℓth segment: To avoid the condition that the data points in a segment only align with the y-axis in the speed–gas plane, **DiVa** performs a PCA analysis and then projects the data points onto the coordinate system defined by the 1st and the 2nd principal components. **DiVa** will then perform a ridge regression to obtain the approximation of GSB curve in the segment. The model parameters recorded for the segments are their 1st and 2nd principal components ($\vec{PC}_{\ell,1}$ and $\vec{PC}_{\ell,2}$) and the regression results.

3.6.6.3 Detection:

Given a testing data $P = (v_{te}, g_{a,te})$, **DiVa** will project P on $\vec{PC}_1$ to identify the segment of the GSB curve P residing in (Fig. 3.8). **DiVa** then computes the distance Δg (Δs) between P and the GSB curve along the gas-axis (speed-axis) and set $d = \min(\Delta g, \Delta s)$, where Δg and Δs are normalized distances to the GSB curve. The physical meaning of this distance is the normalized deviation of speed or gas pedal position of the vehicle that should make a gear shift.

3.7 Evaluation

We first introduce our evaluation setup (Chapter 3.7.1) and then present the evaluation of **DiVa**'s performance from three perspectives. Specifically, we evaluate: i) **DiVa**'s ability to detect anomalies as a pure detection system (Chapter 3.7.3); ii) **DiVa**'s ability to identify the anomaly source(s) while considering the source identification as a stand-alone function (Chapter 3.7.4); and iii) **DiVa**'s system-wide performance as a complete forensic system (Chapter 3.7.5).

3.7.1 Experimental Setup

3.7.1.1 Data

To ensure **DiVa** can be applied to more than one specific vehicle model/make, we use three sources of data for evaluation: i) data collected from 2 vehicles — 1 sedan (Lincoln MKZ) and 1 SUV (Ford Escape) — using commercial off-the-shelf (COTS) OBD-II interfaces, *i.e.*, OpenXC reference VI [98] with 10Hz sampling rate, and ii) Comma AI open dataset [27]. Table 3.3 shows the statistics of the data used in our evaluation. The vehicles' operation scenarios include driving in urban/suburban areas and highways, and do not contain repeated paths in different routes/traces. Also, the vehicles are driven by multiple drivers to avoid the bias of driving behavior.

Source	Total Dist	Trace #	Training	Testing	Driver #
Sedan	55.88 km	7	22.18 km	33.70 km	3
SUV	247.25 km	5	43.76 km	203.50 km	2
Comma	262.07 km	6	16.85 km	245.22 km	Unknown

Table 3.3: The data used in our evaluation (10Hz sampling rate).

While most DSs can generate a detection result using a single data sample, those DSs utilizing correlation coefficients in their detection use a 10s-observation window for detection-by-distance and all DSs use a 100s-observation window for detection-by-model-change. For each data source, our evaluation is based on training and testing steps as follows.

3.7.1.2 Training

DiVa first generates normal behavior models for detection sets directly based on the training data. Instead of using a set of preset thresholds, **DiVa** performs the (optional) threshold training to obtain the necessary parameters for its operation automatically based on the detection preference set by the developers.

One of the most important questions in any detection system is how to set the thresholds for reporting the detection. If a threshold is set to too low, the system will have a high detection rate but will also inevitably have more false positives. On the other hand, if the threshold is set to too large, the system will have less false positives, but more miss detections.

DiVa will automatically generate the training cases and find a threshold combination that minimizes a pre-defined objective function based on the design choice or even system developers' preferences. All a developer needs to do is to set i) the target deviation levels that **DiVa** should consider as an anomaly (*i.e.*, the detection target) and ii) the preference tuple (w). For example, the levels can be set to values smaller than (or equal to) the maximum tolerable error for the vehicle's internal components so that any larger manipulation

which may lead to unstable system state will be considered as an anomaly. Note that this detection target should also consider the behavior change caused by different driving contexts, such as the total mass of the vehicle including cargoes and passengers. For example, there can be a 5.2% mass change (per additional passenger)[4] to the vehicle. The detection target of 1.5m/s^2 for a_X is the result of considering the fact that the vehicle can have at least 400kg payload tolerance (*i.e.*, the driver, passengers, and cargoes) and can be traveling on a road with $\leq$20% road grade. We would like to stress that the target detection settings will not limit **DiVa**'s detection to those target values. As we will introduce next, **DiVa** will automatically adjust the threshold settings to obtain the best TPR and FPR preferences set by the user and these thresholds may be smaller/tighter than the target detection setting; this is the very reason why **DiVa** can still achieve >79% median TPR even if the adversary adjusts the attacks to match the detection settings (Fig. 3.9).

DiVa utilizes the preference tuple $w = [w_1, w_2, w_3]^T$ for each DS to specify the preference for i) true positive rate (TPR), ii) false positive rate of distance to model detection (DFPR or FPR-D), and iii) false positive rate of model change detection (MFPR or FPR-M), respectively. **DiVa** selects the thresholds (Γ) for each DS by minimizing the following objective function:

$$f(\Gamma) = w^T R(\Gamma), \tag{3.29}$$

where $R(\Gamma) = [(1 - TPR(\Gamma))^2, DFPR(\Gamma)^2, MFPR(\Gamma)^2]^T$.

While **DiVa** utilizes always-on DSs to detect the occurrence of an anomaly, always-on DSs and their parameters are chosen to achieve high accuracy (*i.e.*, high detection rate and low false positive rate). On the other hand, since other DSs are utilized to narrow down the search space, their parameters can be chosen to focus on high detection rate. The rationale behind this design choice is to reduce the probability that **DiVa** incorrectly rules out data

[4]Computed based on the average weight of a sedan (1360kg[95]) and 75kg per person: 5.2% = (Per Person Weight) / (Car + Driver).

as the source of anomaly. We choose the default setting of **DiVa** to be (1,1,1) for always-on DSs and (2,1,1) for identification-assistance DSs (*i.e.*, other DSs that are not always-on DSs).

Specifically, **DiVa** adopts 2-fold cross-validation to avoid overfitting the training data, and the basic steps are:

- (Step 1) Create a dataset D_T for threshold selection by performing data manipulation based on the detection target set by system users/developers (see the description after the basic steps);
- (Step 2) Use an optimization algorithm and a half of D_T to find a set of $\{p\}$ that minimizes $f(p)$; and
- (Step 3) Use the second half of D_T to find the specific $p' \in \{p\}$ that minimizes $f(p)$.

In our evaluation, the detection target is set to the "Shift" row in Table 3.4. Therefore, during Step 1, D_T will be randomly manipulated to deviate from its original values by the values specified in Table 3.4. This way we are actually simulating the worse-case scenario that an attacker knows **DiVa**'s detection target and tries to manipulate the data right on that value(s). Since the detection target presented in this section is already a very tight bound, system developers can directly use these settings in their deployment.

In Step 2, we use a genetic algorithm [85] with 400 elements in each generation and as the optimization algorithm and it is set to terminate after the 5th generation to preserve variations between elements. Finally, the top 10 fitting elements will be used in Step 3. The parameters used in the genetic algorithm are chosen based on our preliminary experiment in which thresholds can already converge with this setting. Therefore, picking a larger number of elements of generations will not provide any practical benefit.

If there are some specific vehicle modes or features that will change the vehicle's normal behavior (*e.g.*, the speed capping function or sports mode), the developers should set

the training process to consider those modes by including them in the training data and implementing **DiVa** to have multiple sets of normal behavior models (*i.e.*, **DiVa** should report the detection of an anomaly only when the vehicle does not match any set of the normal behavior models).

α is chosen based on two metrics during the training phase: i) the desired false-positive performance and ii) the maximum tolerable detection latency. Because DiVa's default application is for vehicle diagnostics, say, after a trip, we use i) in our evaluation. Specifically, α is obtained when **DiVa** reaches a 0% median false positive rate.

3.7.1.3 Test Cases

Since **DiVa** is designed to detect vehicle behavior anomalies (not the attack patterns), the test cases are designed (and presented) based on the resulting data/behavior deviation that will be perceived by **DiVa** after an attack is launched (instead of how the attack is launched). That way, we can objectively evaluate **DiVa**'s performance with regard to different behavioral changes without relying on a certain assumption of ⟨controller design, attack⟩, and the results presented here can be extended to predict **DiVa**'s performance under attacks that generate the same post-effect.

Type	g_a	b_r	θ_S	r_m	g_e	a_X	a_Y	ω_Z	v
Unit	%				Lvl.	m/s^2		rad/s	kph
Set	S_1	S_2	S_3	S_4	S_5	S_6	S_7	S_8	S_9
	10				1	1.5		0.5	12
Rand	S_{10}	S_{11}	S_{12}	S_{13}	S_{14}	S_{15}	S_{16}	S_{17}	S_{18}
	[0,100]				[1,6]	[-5,5]		[-5,5]	[0,120]
Shift	S_{19}	S_{20}	S_{21}	S_{22}	S_{23}	S_{24}	S_{25}	S_{26}	S_{27}
	10				1	-1.5	1	0.5	12

Table 3.4: The basic testing scenarios in our evaluation.

Table 3.4 summarizes the basic scenarios considered in our evaluation. Since there is no limit to how anomalous the vehicle behavior can be or how data can be manipulated, we evaluate **DiVa**'s performance while VDTs under consideration can be anomalous in three

representative conditions. These evaluation cases are designed based on the successful attacks reported before. According to [70, 88, 140], there are two steps to incorrectly trigger vehicle functions (*e.g.*, auto parking or diagnostic mode) in modern vehicles — i) setting certain vehicle state data (*e.g.*, g_a and v) to a specific value/range before triggering the vehicle mode and ii) sending a (malicious) command to trigger the vehicle mode. Thus, we first evaluate the cases when any type of data can be set/manipulated (denoted as *Set*) to a certain value (*e.g.*, A3 in Chapter 3.3). Second, we consider the cases when the data can have random values (denoted as *Rand*) within a certain range to simulate sensor failure or a naïve attack (*e.g.*, A2). Finally, we evaluate the cases when any type of data can be shifted by a certain amount of offset, denoted as *Shift*, which simulates conditions in which the attackers move the data or change the ECU/controller behavior to the target range without changing the correlation between VDTs (*e.g.*, A1). *Shift* attacks are also stealthy in that they are controlled to manipulate the vehicle data/behavior right at **DiVa**'s target detection settings (*i.e.*, only a small behavior deviation) and can thus evade the previous detection schemes with only $<$56% detection rate and a long detection time required.

Note we only listed the basic evaluation settings used in Chapter 3.7.3 and other more sophisticated cases will be evaluated in Chapter 3.7.5. For each testing scenario, we generate 10 different test-cases, creating a total of 4,860 and 147,420 different $\langle$trace/route, anomaly$\rangle$ test-cases in Chapters 3.7.3 and 3.7.5, respectively. The anomaly will start at a random time and last up to 5 minutes. The choice of 5min is to facilitate measurements *up to*, but not just for 5min because the metrics used in our evaluation are computed at *every sampling time*, not by considering the entire attack duration as a whole. Also, as we will show later, **DiVa** usually requires only a single data sample to detect an anomaly.

3.7.2 Post-Attack Effects

Unlike drones or aerial vehicles that have a clear definition of unsafe state (e.g., a stall or crashing into the ground), whether or not an attack can cause some real harm to a car

depends on its operating "context" (e.g., whether or not there are obstacles nearby into which the car may crash). This is the reason why we directly presented the resulting effect of attacks. In fact, the behavioral deviation caused by the attacks can be directly inferred from Table 3.4. For example, a 12% vehicle speed deviation may make the driver think s/he is driving at the speed limit while the car actually exceeds the speed limit by 12% or the path planning of autonomous control will have a 12% estimation error for the traveling time to reach a waypoint.

Here we present the real-world effects of the data manipulation presented in Table 3.4. The *upper bound* of perception deviation caused by the manipulation of dynamics data is given by:

- a_X or a_Y manipulation: $\Delta L_{(X \text{ or } Y)} = \int_0^T \int_0^t \Delta a_{(X \text{ or } Y)}(\tau) d\tau dt$, where $\Delta L_{(X \text{ or } Y)}$ is the location deviation in X or Y direction introduced to the vehicle whenever an acceleration manipulation with magnitude $\Delta a_{(X \text{ or } Y)}$ is made, and T is the attack duration;
- v manipulation: $\Delta L = \int_0^T \Delta v(t) dt$, where Δv is the magnitude of speed manipulation; and
- ω_Z manipulation: $\Delta H = \int_0^T \Delta \omega_Z(t) dt$, where ΔH is the vehicle heading deviation and $\Delta \omega_Z$ is the magnitude of yaw rate manipulation.

For example, under a "Shift" attack with $\Delta a_X = 1.5$, the perception deviation of the vehicle's location is bounded by $0.75T^2$ m. As we will show later, **DiVa** achieves 0 median detection gap/latency under "Shift" attacks (*i.e.*, **DiVa** is able to identify the first anomalous data sample in more than 50% of the test cases), so the anomaly will be detected before the vehicle deviates $0.75 \times (0.1)^2$m from its set course most of the time.

Note that since manipulating VDTs of control inputs will not violate the vehicle's reaction to control inputs (*i.e.*, A1 in Chapter 3.3) if the vehicle directly executes the manipulated control inputs, we consider them as pure data manipulation on the IVN and omit their post-attack effects here.

3.7.3 Performance of Detection

3.7.3.1 Metrics

We evaluate the detection performance of **DiVa** using the following metrics:

- *True Positive Rate* (TPR): TPR follows the standard definition of recall that represents the ratio of the number of *data samples* the system correctly found anomalous to the total samples of the anomaly. Note that the miss rate or false negative rate (FNR) = 1 - TPR.
- *False Positive Rate* (FPR): FPR follows the standard definition of fall-out that represents the fraction of samples the system incorrectly reports anomalous. Since **DiVa** utilizes two detection mechanisms with different detection cycles, we report their FPRs individually, including the values of distance-to-model detection (FPR-D) and model-change detection (FPR-M).
- *Detection Gap* (DG): The number of additional data samples after receiving the first anomalous data that **DiVa** requires to report a positive detection and start the false-positive filtering counter. This metric specifies the minimum data count for **DiVa** to be able to detect the anomaly.

Since TPRs/FPRs are computed based on the *sampling time*, they can be considered as the probabilities of true-/false-positives occurring at any given time, and DG is designed to measure how **DiVa** reacts to the length/duration of attacks/anomalies.

3.7.3.2 Baseline Comparison

While we are not aware of any model-based approach that can be directly applied to **DiVa**'s detection scope, we implemented EVAD [54] (*i.e.*, a correlation-based approach that has reported the best performance so far) and PID-Piper [28] (*i.e.*, one of the latest control invariant approaches based on machine learning and cumulative sum (CUSUM) detection without the requirement of physical modeling) for a baseline comparison. For

a fair comparison, EVAD and PID-Piper can only access those VDTs available to **DiVa**. For EVAD, we added three correlation pairs (*i.e.*, $\langle a_Y, v \times \omega_Z \rangle$, $\langle \theta_S, a_Y \rangle$, and $\langle g_e, v/r_m \rangle$) based on EVAD's pair-wise detection mechanism to ensure it covers the same detection scope as **DiVa**. For PID-Piper, a_X, a_Y, and ω_Z are the prediction outputs of PID-Piper's feed-forward control (FFC) module while g_a, b_r, and θ_S are those of PID-Piper's feedback control (FBC) module. For both modules, the VDTs that are not their prediction outputs are their inputs.

3.7.3.3 Results

Fig. 3.9 summarizes **DiVa**'s detection performance when one VDT is anomalous at a time (Table 3.4). In general, **DiVa** is able to achieve $\geq$93.03% TPR-All (*i.e.*, the average of median TPRs from all three attack types) and low (close to 0%) median FPR-D for all data sources. The slightly lower TPR in individual attack results for detecting a *Shift* data anomaly is due to our assumption of strong attacks under which the attacker knows **DiVa**'s detection targets and manipulates the vehicle's behavior at those targets to evade **DiVa**'s detection. The results of SUV show a higher FPR-M (10.53%) is due to the activation of speed capping function in some of the collected traces which were detected by **DiVa** as anomalies. If such traces are excluded as shown in SUV*, the median FPR-M decreases to 0%. This result actually showcases **DiVa**'s ability to detect changes of the vehicle's real-world behavior which will be discussed more in Chapter 3.7.5. Since the acceleration and yaw rate of *Comma AI* are collected using external IMUs, the change of IMU positions may affect the performance of individual DSs. We conjecture the change of position of IMUs happened in the data set, thus leading to a higher FPR-M (and lower TPR-Shift) in this case. This assumption is based on the fact that the FPR-Ds and TPRs, except for the *Shift* case, are similar to those of the data we collected.

Figs. 3.10 and 3.11 show the detection performance of EVAD and PID-Piper, respectively. One can observe that EVAD can only achieve moderate performance with $\leq$63.91%

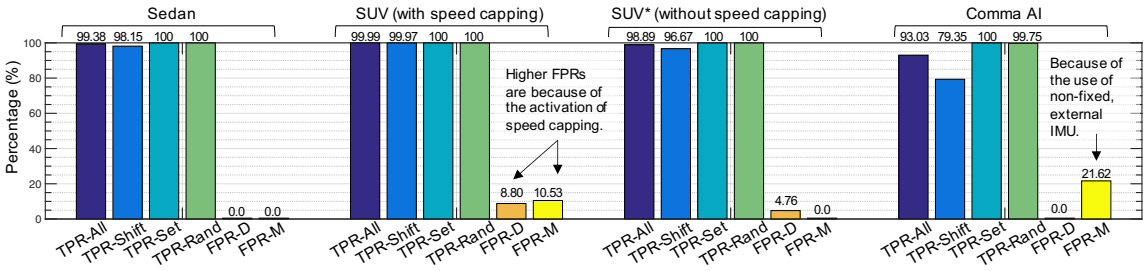


Figure 3.9: **DiVa**'s detection performance, where the TPRs and FPRs are presented using their median values.

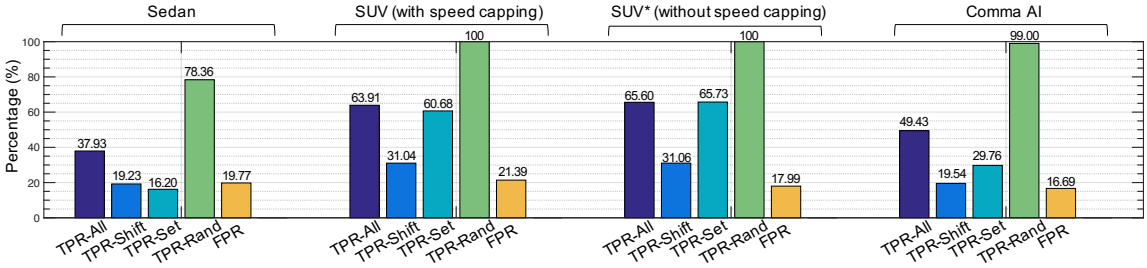


Figure 3.10: EVAD's detection performance, where the TPRs and FPRs are presented using their median values.

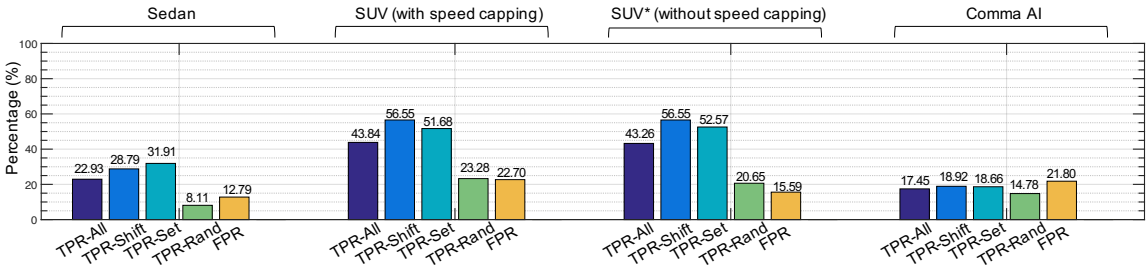


Figure 3.11: PID-Piper's detection performance, where the TPRs and FPRs are presented using their median values.

TPR-All with high (median) FPR ($\geq$16.99%). This is due to EVAD's inaccessibility of strongly correlated data pairs of target setpoints and vehicle dynamics, hence performing detection based on loosely-correlated pairs of commonly-available data. Even though PID-Piper shows a better performance than EVAD in detecting stealthy *Shift* attacks, it can only achieve $<$57% median TPR and $>$12% median FPR, *i.e.*, it cannot capture vehicle behavior as efficiently as **DiVa** under the same training and testing condition. In contrast, **DiVa** is equipped with efficient models to capture vehicle behaviors, achieving 79–100% median TPR and 0–4.76% median FPR-D.

We now compare the required numbers of data samples for **DiVa**, EVAD, and PID-Piper to report the first (true-)positive detection (Table 3.5). **DiVa** is able to achieve 0 median DG in all cases (*i.e.*, **DiVa** can detect the anomaly with only the very first anomalous data most of the time) while EVAD has $\geq$27 best-case median DGs and PID-Piper requires up to 175 median DGs to trigger a positive detection due to their reliance on the correlation coefficients and CUSUM detection. In summary, **DiVa** is shown to have significant advantages over EVAD and PID-Piper in the case of limited data availability thanks to its efficient models and detection mechanisms.

Data	Approach	Median DG				75th Percentile DG			
		All	Shift	Set	Rand	All	Shift	Set	Rand
Sedan	DiVa	0	0	0	0	18	53	0	0
	EVAD	64	87	61	44	159	183	159	134
	PID-Piper	3	6	3	1	31	44	40	8
SUV	DiVa	0	0	0	0	8	23	0	0
	EVAD	58	38	32	105	188	199	199	165
	PID-Piper	14	26	14	3	38	45	41	27
SUV*	DiVa	0	0	0	0	29	87	0	0
	EVAD	86	128	103	27	199	199	199	199
	PID-Piper	76	27	27	175	129	48	65	275
Comma	DiVa	0	0	0	0	47	128	0	13
	EVAD	47	70	41	31	131	173	122	99
	PID-Piper	2	3	2	0	11	14	17	3

Table 3.5: The median and 75th-percentile detection gap (DG) in sample counts of **DiVa**, EVAD, and PID-Piper.

Tables 3.6 and 3.7 show more detailed statistics of performance of each detection set and **DiVa**'s first stage performance. The 25th and 75th percentiles of each metrics are shown along with their means and medians. For the identification-assistance DSs (*i.e.*, the non-always-on DSs), they achieve a high average detection rate (90+%) but also higher FPRs than the always-on DSs because of **DiVa**'s design choice of emphasizing the detection rate. Note that higher FPR (10+%) of identification-assistance DSs will not affect the system-wide false positive rate, because anomaly occurrences are detected by always-on DSs, and the identification-assistance DSs are used for pinpointing the anomalous data.

Since the identification-assistance DSs perform event-driven detection and their DGs depend highly on the timing of the target events (*e.g.*, the vehicle performs a gear shift), we

omit their DGs here.

3.7.4 Performance of Source Identification

3.7.4.1 Evaluation Metrics

Let us evaluate **DiVa**'s identification of anomaly sources (Chapter 3.4.4.2) as a stand-alone function by exploring how well **DiVa** ranks the potential anomaly source(s):

- *System Identification Index* (SII): The average maximum ranking of the anomaly source(s) in the output of **DiVa**. For example, if there are two anomaly sources and the output of **DiVa** suggests that they are ranked 1 and 4, the maximum ranking in this case will be 4. Ideally, the ranking of the anomaly source(s) should be higher (*i.e.*, having a smaller value) than other VDTs.
- *Search Range* (SR): The search range of the anomaly source(s). It is defined as the number of VDTs with the same as, or lower rank than the maximum rank of the anomaly source(s). Using the same example as shown earlier, if there are two sensors ranked 4, SR in this case will be 5. We use ASR to indicate the average SR.

The reason for adopting two different metrics in this case is to identify whether **DiVa** tends to include other VDTs as the anomaly sources with the same rank as the real one(s). Furthermore, (SII - n), where n is the number of manipulated data, is equivalent to how many "false-positive" components are identified as an anomalous source before the actual anomaly source is identified.

3.7.4.2 Results

To evaluate **DiVa**'s performance in locating anomaly source(s) as a stand-alone function, we control the input from the detection stage to have different levels of performance (*e.g.*, TPR and FPR) instead of directly using the results in Chapter 3.7.3. That way, we are able to evaluate **DiVa**'s identification performance under different conditions. We evaluated

Data	Scenarios	TPR (%)				FPR-D (%)				FPR-M (%)				Detection Gap (Data Count)			
		Mean	Median	25th	75th	Mean	Median	25th	75th	Mean	Median	25th	75th	Mean	Median	25th	75th
Sedan	Performance Summary of Individual Detection Set																
	DS1	88.72	99.83	88.60	100	0.25	0	0	0	4.29	0	0	0	139.5	0	0	48
	DS4	93.08	100	93.34	100	3.67	0	0	2.78	2.85	0	0	0	8.75	0	0	0
	DS10	77.09	86.65	63.18	99.27	19.10	16.67	2.86	28.57	1.13	0	0	0	22.24	1	0	125
	DS11	79.05	96.38	66.64	100	0	0	0	0	3.02	0	0	0	21.95	0	0	196
	1st Stage																
	Summary	93.09	100	95.45	100	6.91	0	0	10.96	10.65	0	0	25	89.94	0	0	0
	Shift	88.15	98.15	83.31	100									116.22	0	0	53
	Set	98.04	100	99.13	100									37.21	0	0	0
	Rand	93.09	100	98.87	100									116.39	0	0	0
SUV	Performance Summary of Individual Detection Set																
	DS1	87.19	100	86.95	100	14.31	14.80	9.52	22.94	12.24	15.79	0	20.93	82.2	0	0	0
	DS4	97.66	100	100	100	0.42	0	0	0.46	4.63	0	0	4.65	20.6	0	0	0
	DS10	89.66	100	93.34	100	6.95	5.73	4.59	7.35	10.87	13.95	0	20.00	57.1	0	0	0
	DS11	90.12	100	99.00	100	0.18	0	0	0	11.40	10.53	2.33	20.00	86.4	0	0	0
	1st Stage																
	Summary	91.98	100	97.60	100	8.93	8.80	2.388	16.28	12.63	10.53	0	18.60	87.06	0	0	0
	Shift	81.97	99.97	68.04	100									189.07	0	0	23
	Set	99.38	100	100	100									9.54	0	0	0
	Rand	94.58	100	99.77	100									62.57	0	0	0
SUV*	Performance Summary of Individual Detection Set																
	DS1	86.75	100	93.34	100	11.02	11.73	0	14.80	4.59	0	0	10.53	97.9	0	0	0
	DS4	97.80	100	100	100	0	0	0	0	6.30	0	0	25.00	16.2	0	0	0
	DS10	88.88	100	86.64	100	7.47	5.10	0	19.05	7.15	0	0	21.05	60.7	0	0	0
	DS11	91.37	100	99.40	100	0.02	0	0	0	11.74	10.53	0	25.00	89.6	0	0	0
	1st Stage																
	Summary	91.06	100	96.63	100	7.02	4.76	0	7.65	2.97	0	0	5.26	100.96	0	0	0
	Shift	80.13	96.67	66.64	100									205.88	0	0	87
	Set	99.14	100	100	100									12.19	0	0	0
	Rand	93.91	100	99.14	100									84.82	0	0	0
Comma	Performance Summary of Individual Detection Set																
	DS1	85.87	99.70	92.90	100	0.97	0.53	0	1.67	15.97	13.51	11.11	22.22	95.3	0	0	10
	DS4	89.56	100	96.32	100	2.33	1.59	0	3.33	8.82	10.81	0	12.50	73.5	0	0	0
	DS10	88.71	99.97	92.94	100	14.21	4.35	2.41	26.80	15.62	12.50	11.11	25.00	103.4	0	0	9
	DS11	69.68	95.70	40.19	100	8.02	6.67	4.12	9.30	10.80	11.11	0	16.67	214.2	0	0	86
	1st Stage																
	Summary	80.63	99.70	70.00	100	11.81	0	0	22.68	23.02	21.62	8.33	33.33	200.56	0	0	11
	Shift	62.73	79.35	25.03	100									382	0	0	128
	Set	95.06	100	99.12	100									40.42	0	0	0
	Rand	84.11	99.75	78.30	100									179.26	0	0	13

Table 3.6: **DiVa**'s performance results of always-on detection sets and 1st-stage detection. Note that "Summary" represents the statistics when considering all three attack scenarios as a whole and it is not the same as "All" in Figs. 3.9.

	DS	TPR (%)	FPR-D (%)				FPR-M (%)			
			Mean	Median	25th	75th	Mean	Median	25th	75th
Sedan	2	99.21	0.47	0	0	0	8.85	0	0	14.29
	3	100	3.09	0	0	2.86	13.35	14.29	0	25.00
	5	100	35.26	35.57	27.14	50.00	15.27	14.29	0	28.57
	6	100	1.40	0	0	2.62	6.70	0	0	0
	7	99.05	13.63	11.11	0	25.00	0	0	0	0
	8	97.94	29.81	29.94	14.29	45.21	0	0	0	0
	9	100	8.75	8.33	2.08	13.89	15.75	6.25	0	28.57
	12	99.68	11.17	11.43	0	18.75	2.66	0	0	0
	avg	99.49	12.95	12.05	5.44	19.79	7.82	4.35	0	12.05
SUV	2	97.78	14.43	14.50	11.22	20.41	24.81	26.32	0	39.53
	3	96.89	11.57	12.50	7.14	15.83	7.47	0	0	15.00
	5	98.67	5.29	4.76	1.96	7.14	17.96	16.28	7.63	20.00
	6	97.00	13.06	13.12	8.33	18.18	8.46	10.00	0	15.79
	7	91.56	6.89	5.88	4.59	8.94	0	0	0	0
	8	91.56	10.47	12.24	6.86	14.80	0	0	0	0
	9	85.33	19.51	17.20	11.76	33.16	8.67	6.98	0	15.00
	12	96.67	2.23	0.69	0	5.39	3.53	0	0	5.26
	avg	94.43	10.43	10.11	6.48	15.48	8.86	7.45	0.95	13.82
Comma	2	97.22	0.91	0.53	0	1.67	20.40	22.22	16.67	25.00
	3	74.26	0	0	0	0	1.01	0	0	2.70
	5	94.72	12.70	10.87	9.14	16.88	9.95	8.33	0	12.50
	6	96.39	12.25	13.18	10.47	15.22	29.86	33.33	23.61	37.84
	7	83.33	5.60	5.42	3.33	6.67	0	0	0	0
	8	96.85	40.08	31.81	26.74	58.70	0	0	0	0
	9	87.78	13.31	10.15	6.19	22.04	22.67	16.22	12.50	33.33
	12	90.56	16.13	12.50	10.31	21.67	29.33	33.33	16.22	37.50
	avg	90.14	12.62	10.56	8.27	17.86	14.15	14.18	8.63	18.61

Table 3.7: Performance results of identification-assistance DSs.

the scenarios when n (=1$\sim$4) VDTs are manipulated while using the traces collected from the vehicles to simulate the time when certain events like gear shifts occur. Similarly to the evaluation in Chapter 3.7.3, we tested every case 10 times while all the anomalies start randomly at the same time between the start and the end of each trace. All combinations for given n are tested in our evaluation, *i.e.*, 9, 36, 84, and 126 combinations are tested for n=1, 2, 3, and 4, respectively.

TPR (%)	FPR (%)	n=1		n=2		n=3		n=4	
		SII	ASR	SII	ASR	SII	ASR	SII	ASR
100	0	1.00	1.00	1.22	3.09	1.36	4.82	1.56	6.44
90	10	1.03	1.04	2.44	2.86	4.00	4.67	5.53	6.33
80	20	1.14	1.17	2.91	3.16	4.70	5.05	6.26	6.64

Table 3.8: **DiVa**'s identification performance.

Table 3.8 summarizes the results of SIIs and ASRs. When n=1, **DiVa** is able to pinpoint

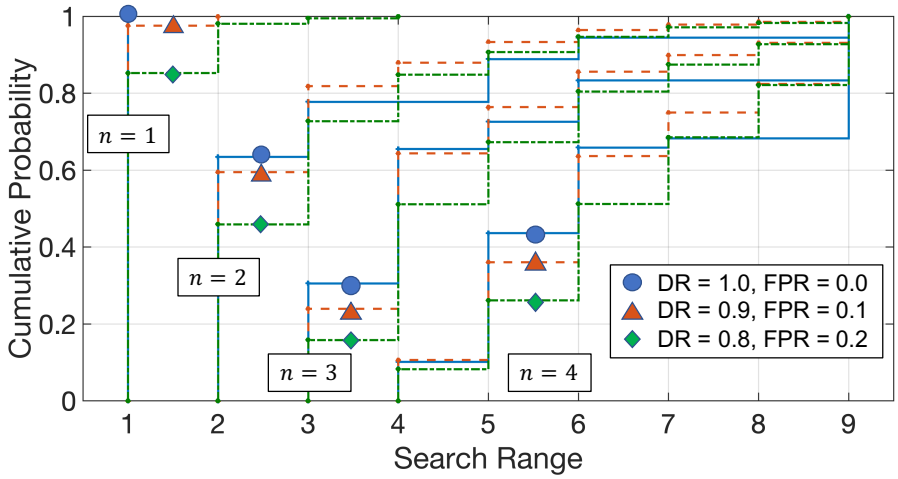


Figure 3.12: CDF of search space under different attack scenarios, where DR represents detection rate (i.e., true positive rate or TPR).

the source of anomaly (SII≈1, ASR≈1) even when the detection has only 80% TPR and 20% FPR. When n=2, **DiVa** typically narrows down the search space to 2–3 data. Theoretically, it will need $C^9_{9-n} = C^9_7 = 36$ DSs to exhaustively test all the cases when 2 data are manipulated. As n increases to 3 and 4, the search range becomes to 4–5 and 6–7, respectively. The results show that **DiVa** has better performance when 1 or 2 components are anomalous and can still narrow down the search space even when close to half of the data are anomalous.

Fig. 3.12 further shows the CDF of DiVa's performance in narrowing down the search space of the anomaly sources when there can be different numbers of manipulated data (n) and detection performance. Specifically, DiVa can still narrow down the search space to 6 data in more than a half of the test cases when there are 4 manipulated data even if the detection sets have a moderate (80%) TPR and a high (20%) FPR.

3.7.5 System-wide Performance

Besides the metrics used in Chapter 3.7.4, we include *System Detection Gap* (SDG) as a metric. Like DG, SDG is the number of (additional) data between the first anomalous data and the one that triggers **DiVa** to lock on the source.

3.7.5.1 Single Data Manipulation

Table 3.9 shows **DiVa**'s performance when there is only one anomalous data. We first compare the performance of different data sources. While the average SIIs of *Sedan* (1.38) and *Comma* (1.53) are less than 2, that of *SUV* (2.35) is significantly larger than the other two sources. This result indicates **DiVa**'s ability to identify the exact anomaly source for *Sedan* and *Comma* in most of the time, but **DiVa** tends to rank another data higher than the actual anomaly source in case of *SUV*.

	Test Cases	SII	Search Range (SR)				SDG			
			ASR	Med.	25th	75th	Avg.	Med.	25th	75th
Sedan	Set	1.14	1.24	1	1	1	64.3	0	0	36
	Rand	1.41	1.56	1	1	2	246.9	0	0	338
	Shift	1.58	1.73	1	1	2	204.4	19.5	0	199
	Avg	1.38	1.51	1	1	1	171.9	6.5	0	191
SUV	Set	2.10	2.24	2	1	4	13.1	0	0	0
	Rand	2.34	2.50	2	1	4	49.6	0	0	0
	Shift	2.60	2.73	2	1	4	180.5	0	0	28
	Avg	2.35	2.49	2	1	4	81.1	0	0	9.3
SUV*	Set	1.03	1.26	1	1	1	10.9	0	0	0
	Rand	1.57	1.73	1	1	2	45.5	0	0	0
	Shift	1.80	1.96	1	1	2	212.5	0	0	47
	Avg	1.47	1.65	1	1	1.67	89.6	0	0	15.7
Comma	Set	1.10	1.50	1	1	2	35	0	0	16
	Rand	1.27	1.59	1	1	2	87.8	0	0	27
	Shift	2.23	3.43	2	1	4	775.3	139	0	1219
	Avg	1.53	2.17	1.33	1	2.67	299.4	46.3	0	420.7

Table 3.9: **DiVa**'s performance when one data/behavior is manipulated.

As mentioned earlier, the reason for this is that during some trips, the speed capping function of *SUV* was activated (*i.e.*, the vehicle will not exceed 75 mph even if the driver presses the gas pedal). Since we use the data without speed capping for training, **DiVa** identifies the speed capping as an anomaly. If we remove the traces with speed capping (denoted by *SUV**), the average SII reduces to 1.47, which is very close to those of other data sources. This result not only corroborates our assumption of the cause of speed capping, but also demonstrates **DiVa**'s ability to capture abnormal behavior from the training data in a practical real-life example. This phenomenon can also be observed by comparing the SRs of *SUV* and *SUV**. In what follows, we will consider the results of *SUV** (instead

of *SUV*) as the main results in the evaluation. 7 out of 9 data–attack combinations achieve 0 median SDGs, meaning that **DiVa** is able to "lock on" the anomalous source based on the very first anomalous sample.

Now, we compare the search ranges (SRs) between different attack types. Since *Set* and *Rand* will violate the correlation between VDTs, **DiVa** will have no difficulty in detecting them. In 8 out of 9 data–attack combinations, the medians of search ranges are 1, showing **DiVa**'s abilty to identify the source accurately most of the time in practical scenarios. Specifically, **DiVa** is able to identify the exact anomaly source with 0.91 probability and narrow down the anomaly space to an average of 1.33 VDTs when a *Set* anomaly occurs. Even in the shifting attack case, **DiVa** can still identify the anomalous data with 0.75 probability as the exact anomaly source.

3.7.5.2 Multi-Data Manipulation

Let us consider the case when an attacker knows **DiVa**'s operation and tries to evade **DiVa**'s identification by manipulating multiple data simultaneously. Table 3.10 summarizes the results when $\geq$2 data are manipulated to certain values simultaneously (based on combinations of $S_1 - S_9$ in Table 3.4). Other than slightly higher ASRs of *Comma* in n=2 cases due to the higher FPR-Ms as mentioned in Chapter 3.7.3.3, **DiVa** can identify anomalies close to the theoretical value (90% TPR in Table 3.10), *i.e.*, **DiVa** has good performance in detecting and identifying generic multi-data manipulation attacks.

	n=2			n=3			n=4		
	SII	ASR	SDG	SII	ASR	SDG	SII	ASR	SDG
Sedan	2.57	2.98	16	4.10	4.73	6.2	5.49	6.26	1.8
SUV*	2.52	3.42	30	3.65	5.18	0.9	4.64	6.67	0.7
Comma	4.00	4.25	78.7	5.59	5.91	2.4	6.79	7.13	0.9

Table 3.10: **DiVa**'s performance when multiple data are anomalous.

3.7.6 Replay Attacks

We further assume a strong attacker who (1) knows the design of **DiVa**, the result of **DiVa**'s model training, and the data covered by each DS and (2) directly targets the data in a certain detect set to evade its consistency check. Once the attacker chooses a target DS, all the VDTs covered by the target DS will be replaced by the recorded values from the same vehicle (*i.e.*, a replay attack) and, therefore, the correlation between the replaced data will perfectly match the vehicle's normal operation. Table 3.11 shows the detection and identification results of replay attacks. **DiVa** is shown to be able to achieve high median TPR ($\geq$ 91.75% for sedan and SUV*) and 0 DG in all but one tested settings, *i.e.* **DiVa** has no difficulty in capturing a sophisticated replay attack when the attacker utilized replay attacks to target certain detection sets to evade **DiVa**'s detection. Note that because a replay attack will only trigger weak detection as defined in Chapter 3.4 and the data will be divided into two groups that perfectly preserve the vehicle's normal behavior, **DiVa** will have higher search range or even include all the data as potential anomaly sources.

Tables 3.12 and 3.13 summarize EVAD's and PID-Piper's performance when the attacker manipulates the same VDTs as Table 3.11, respectively. Note that since EVAD and PID-Piper utilize different data pairs/groups from **DiVa** to perform its detection, these attacks will actually give advantage to EVAD/PID-Piper when we compare their performances to **DiVa**. However, one can observe that **DiVa** can still achieve similar or up to 46.5% and 82.14% increase of absolute (median) TPRs with much smaller FPRs than EVAD and PID-Piper even if the attacks are specifically designed to evade **DiVa**'s (neither EVAD's nor PID-Piper's) detection, respectively.

Data	Targeted DS Data	TPR (%)				SII	Search Range				Detection Gap (Data Count)			
		Mean	Median	25th	75th		ASR	Median	25th	75th	Mean	Median	25th	75th
Sedan	DS1	85.06	96.55	72.46	100	5.27	7.29	7	6	9	298.09	0	0	304
	DS4	50.45	58.51	32.29	66.92	5	8.59	9	9	9	435.92	109	0	763
	DS10	86.68	97.33	70.01	100	7.16	8.97	9	9	9	285.74	0	0	240
	DS11	86.53	98.07	72.85	100	6.9	8.93	9	9	9	316.99	0	0	249
	Summary	77.18	91.75	62.69	100	6.08	8.44	9	9	9	334.18	0	0	510
SUV*	DS1	94.79	100	93.50	100	5.8	7.6	8	6	9	6.15	0	0	0
	DS4	45.87	48.32	28.23	60.63	3.8	9	9	9	9	273.47	0	0	191
	DS10	98.07	100	99.45	100	7.7	9	9	9	9	0.7	0	0	0
	DS11	93.84	98.93	93.49	100	7.6	9	9	9	9	20.7	0	0	0
	Summary	83.14	98.67	76.71	100	6.23	8.65	9	9	9	75.26	0	0	0
Comma	DS1	57.80	55.91	33.09	89.74	5.83	8.67	9	9	9	348.61	0	0	278
	DS4	57.15	58.33	37.02	79.99	5.43	9	9	9	9	260.22	0	0	81
	DS10	90.37	100	86.67	100	7.82	8.93	9	9	9	127.05	0	0	0
	DS11	58.06	60.00	34.84	81.79	7.18	8.82	9	9	9	310.43	0	0	416
	Summary	65.84	69.89	40.59	95.52	6.57	8.85	9	9	9	261.58	0	0	65

Table 3.11: **DiVa**'s 1st-stage performance under targeted replay attacks, where the FPRs are the same as the "1st Stage" shown in Table 3.6.

Data	Targeted DS Data	TPR (%)				FPR (%)				Detection Gap (Data Count)			
		Mean	Median	25th	75th	Mean	Median	25th	75th	Mean	Median	25th	75th
Sedan	DS1	71.97	75.32	60.88	90.20					295.19	179.5	0	587
	DS4	76.00	74.62	68.72	100					377.81	247	0	754
	DS10	86.01	89.25	75.44	100	32.26	27.51	18.58	42.74	359.87	281.5	0	700
	DS11	84.09	84.87	71.41	100					310.3	217.5	0	612
	Summary	79.52	79.70	70.47	100					335.79	238	0	673
SUV*	DS1	80.29	89.97	71.74	100					89.85	0	0	0
	DS4	96.07	100	95.45	100					108.9	0	0	82
	DS10	96.10	100	95.58	100	31.94	26.71	23.07	42.68	93.7	0	0	2
	DS11	96.06	100	100	100					114	0	0	0
	Summary	92.13	100	90.75	100					101.61	0	0	2
Comma	DS1	27.94	22.71	10.98	46.15					499.17	499	378	499
	DS4	4.36	0.13	0.13	6.12					553.27	499	499	499
	DS10	34.20	31.62	23.39	44.39	30.87	25.92	7.16	48.72	460.08	499	277	500
	DS11	39.39	38.70	23.18	55.25					419.22	492	280	499
	Summary	26.47	23.39	6.12	44.39					482.94	499	360	499

Table 3.12: EVAD's detection performance under targeted replay attacks.

3.8 Discussion

3.8.1 Deployment and Limitations:

Since **DiVa** is designed as a software-based solution, it can be flexibly deployed within vehicles as long as it can access the VDTs. Due to **DiVa**'s flexibility, it can be directly deployed as an aftermarket or add-on solution for modern vehicles (*e.g.*, the same method we conducted the evaluation with an OBD-II interface). It can also be placed in an in-vehicle network gateway or integrated into the existing ECU modules as a first-party solution.

Data	Targeted DS Data	TPR (%)				FPR (%)				Detection Gap (Data Count)			
		Mean	Median	25th	75th	Mean	Median	25th	75th	Mean	Median	25th	75th
Sedan	DS1	20.70	14.32	9.46	20.79	18.02	12.84	9.22	25.60	261.07	15	2	129
	DS4	29.27	28.96	18.95	36.69					196.17	26	4	155
	DS10	29.18	29.28	21.12	38.69					324.39	61	3	247
	DS11	27.22	27.71	17.71	35.59					259.73	42	2	114
	Summary	26.60	25.69	15.23	35.22					260.34	38	3	143
SUV*	DS1	12.31	13.66	2.48	20.94	33.30	33.44	18.06	56.64	319.3	27	16	489
	DS4	11.97	7.01	0.88	16.03					525.87	171	56	1074
	DS10	27.93	25.34	7.86	41.95					85.25	0	0	222
	DS11	21.03	19.39	9.41	29.52					71.85	3	0	148
	Summary	18.31	16.53	5.51	25.34					250.57	27	0	276
Comma	DS1	27.74	29.69	21.83	34.86	23.83	21.81	19.04	28.07	15.4	0	0	6
	DS4	15.37	14.13	11.84	19.45					33.33	0	0	31
	DS10	26.31	25.09	19.36	32.94					15.2	0	0	2
	DS11	24.10	22.29	17.63	30.89					45.63	0	0	5
	Summary	23.38	21.84	14.34	30.77					27.39	0	0	6

Table 3.13: PID-Piper's detection performance under targeted replay attacks.

Besides the detection limitation discussed in Chapter 3.3, there is still room for improvement, such as finding the optimal time to react to a positive detection to avoid false-positives from affected vehicle's normal operation, if **DiVa** is to be used as a real-time defense against safety-critical attacks. For example, if i) an anomaly is detected by model change but not from detection by distance to model, ii) it is not from any of the control input (CI), and iii) the deviation from the normal behavior is mild, it does not require an immediate action since the anomaly has a high probability to have been caused by a component wear-out. On the other hand, if the anomaly is detected by both detection mechanisms and the anomalous component is in the scope of CI, **DiVa** should immediately notify the driver and stop using any autonomous function related to the anomalous component.

3.8.2 Contexts and Operation Modes:

DiVa can automatically incorporate common operation contexts (e.g., acceleration deviations caused by road inclinations) into its threshold settings and make new adjustments by training on the data from the entities's daily routines right after a regular maintenance. For different operation modes, **DiVa** can create different models for the same DS with the training data divided by i) outlier tests (with limited data availability) or ii) operation-mode data types (if available). **DiVa** can then report detection of anomalies only when no model

matches the entity's behavior. For example, it can be updated to account for i) road inclinations by adding a calibration term in DS_1-DS_3 if inclinometer/elevation data is available and ii) tire slippage by identifying the features of insufficient friction embedded in acceleration measurements.

3.9 Conclusions

We have proposed **DiVa**, a new forensic tool for SA systems. **DiVa** operates on multiple detection sets with overlapping data. **DiVa** can cross-validate the data and narrow down the search space to identify the source(s) of anomaly. Our extensive experimentation based on commercial vehicles has shown **DiVa** to have significant advantages over prior work in the usual case of limited data availability and pinpoint the anomalous data with 0.75 probability even in the case of sophisticated single-data manipulation. Furthermore, **DiVa** can narrow down the search space for an anomaly source to an average of 1.33 sensors, and pinpoint the exact anomalous data with 0.91 probability under a set-to-value attack.

CHAPTER IV

EDRoad: Easy-to-Use System for Received Data Verification

4.1 Introduction

SA systems or mobile Cyper-Physical Systems (CPSes) in general, such as ground and air vehicles, are the most common, representative systems we rely on everyday. To run existing and emerging applications, communication interfaces have been integrated into mobile CPSes and will be exploited to provide advanced functionalities in future CPS services. For example, AERIS program (subcategory ECO-Signal Operation [134]) of the U.S. Department of Transportation specifies a future Intelligent Transportation Systems (ITSes) to perform smart traffic light control and provide maneuver/routing advice to individual (autonomous) vehicles based on their reported locations via vehicular communications and the timing information from traffic signal controllers for enhancing the smoothness of travel and reducing fuel consumption (see Fig. 4.1). To support such ITS services, the infrastructure, such as roadside units (RSUs) and traffic light controllers, must monitor vehicles' status/behavior on the road, especially their location, speed and heading, via standardized messages like Basic Safety Messages (BSMs) from the vehicles on the road and/or other sensors.

The integrity of data or information received from mobile CPSes like cars, especially

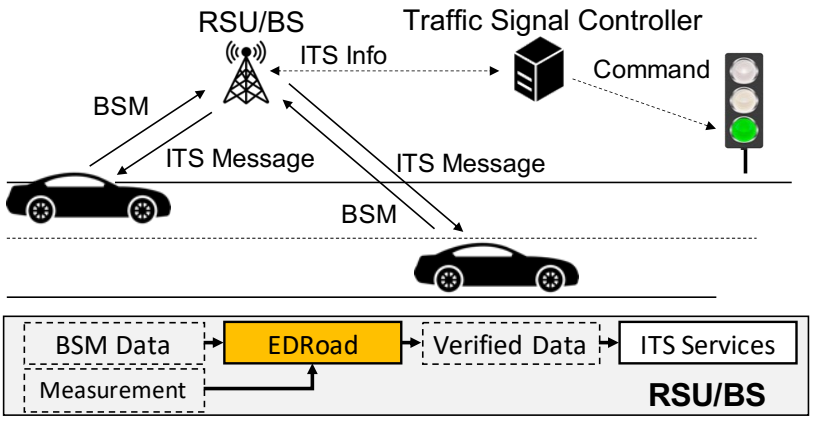


Figure 4.1: An example of **EDRoad**'s application scenario in a typical ITS setup adapted from [134].

the dynamic measurements, is critically important for the infrastructure to produce correct output for CPS services. For example, prior work has shown that a larger discrepancy between the actual and the perceived/received vehicle information may have more severe influence on the ITS and can even completely negate the benefits brought by the ITS — a single vehicle can cause a 68% increase of average waiting time at an intersection with a smart traffic light controller [19]. Therefore, to ensure CPS services function correctly, there must be an efficient way to verify if the received information is trustworthy. Such a system should *detect* the existence of anomalous data (*i.e.*, data that exceed a specified level of error-tolerance), *identify* which data is anomalous, and *restore* the anomalous data because simply discarding the anomalous data (or all received data) cannot help CPS services capture the status of the target CPS (*e.g.*, the vehicle location/speed/heading in the aforementioned ITS example).

One of the most important CPS service properties is that the operation environment/requirement of the infrastructure and mobile CPSes may vary with location and time, and their hardware can also be continually upgraded to support emerging applications. For example, because there is no universal standard on which sensors should be available in RSUs and RSUs may be deployed to support various services and applications, different RSUs will very likely need to verify different data types and sensor measurements. Fur-

thermore, RSUs may also experience constant hardware upgrades, such as adding a new set of cameras for traffic monitoring. However, existing anomaly detection systems with the capability of determining which of the received data exceeds the error-tolerance level (instead of directly treating all data as a *single* group) are usually tailored for a specific system architecture (*i.e.*, assuming a fixed set of data and verification formulations/procedures). Therefore, they cannot easily expand their detection scope or adapt to different deployment scenarios (Chapter 4.2).

To the best of our knowledge, there still does not exist any efficient way to set up a system for detecting and restoring *each* anomalous data without requiring engineers to manually design i) individual detectors/observers for their data of interest and ii) the verification process, including how (and in what sequence) to identify and restore the anomalous data (Chapter 4.2). This leads to an important question: *Is there any efficient way to help inexperienced engineers*[1] *develop a system at design time for detection and recovery of individual anomalous data while handling varying application requirements without manually constructing/tailoring the verification process?*

To answer this question, we propose **EDRoad**, a system and design concept that is capable of *detecting* and *recovering* the *individual* anomalous data received from external entities. Fig. 4.1 shows an application example, where **EDRoad** acts as a data corrector in an RSU or a base station (BS) that corrects the anomalous data in BSMs received by the RSU/BS to prevent their use in ITS services.

From a technical perspective, **EDRoad** can be viewed as a novel ensemble learning framework tailored for *multi-data* verification. That is, given a group of data, **EDRoad** will identify which of them are potentially anomalous (*i.e.*, exceeding a specified level of error tolerance) and, if anomalous, restore their correct values. It automatically constructs observers/classifiers that monitor the correlation between its data of interest and combines the classification results to identify which data are anomalous and then restore them. Specif-

[1]Engineers with the domain knowledge of the correlation/causality between data but not how to design a system for data integrity check.

ically, it greedily maximizes the lower bound of average detection rate and minimizes the upper bound of average false-positive rate in identifying *individual* anomalous data. Its design enables the following three key features:

F1. *Easy Deployment.* **EDRoad** is designed to assist engineers with little to no knowledge of how to construct a data verification system (*i.e.*, **EDRoad**'s users need not construct anomaly detectors/classifiers by themselves). All an engineer needs to do is to provide i) the correlation or causality between the data to be verified, ii) the noise distributions under normal operation, and iii) raw data traces for training purposes, *i.e.*, engineers need not manually adjust the training cases (Chapter 4.4.3). There are only two deployment requirements for **EDRoad**. First, there must exist a measurement/data that has bounded errors when it is utilized to observe one or more target data d_T to be verified. Second, every data to be verified is correlated with some d_T or another data.

F2. *Easy Expansion.* To expand or modify **EDRoad**'s detection scope, all engineers need to do is to identify the cyber-physical correlation/causality between the new sensor/-data and the existing ones without (re)designing the detailed detection mechanism and the knowledge of anomaly detection. For example, if an engineer wants to verify vehicle acceleration (a) in addition to the original design that only verifies vehicle speed (v), s/he only needs to include the formulation $v_{k+1} = v_k + a_k \Delta t$ in **EDRoad**, where k is the time index and Δt is the sampling interval, and **EDRoad** will handle the rest of integration.

F3. *Easy Configuration.* **EDRoad** provides a straightforward (optional) mechanism for engineers to set the error-tolerance level (*i.e.*, detection threshold) for *individual data* that matches their need. This feature may appear to be basic, but most prior approaches do not provide any mechanism/mapping to adjust their thresholds since they utilize layers of feature extraction (*e.g.*, Power Spectral Density [54]) to perform detection that is no longer directly associated with the original data.

Specifically, **EDRoad** takes a set of system descriptions that depict the domain-knowledge of the target system/data (Chapter 4.4) and optional error-tolerance as inputs and transforms

them into their corresponding classifiers. It further combines the classifiers to generate a set of potential verification procedures and models to follow during its online execution. Finally, **EDRoad** performs data verification and recovery while dynamically adjusting its procedure and models to match its operating environment. In this chapter, we use BSM verification in an RSU with vehicle-to-infrastructure (V2I) communication as a concrete case study to illustrate **EDRoad**'s design (Fig. 4.1). However, in addition to assisting monitor-based services (*e.g.*, reliable surveillance for ITSes [159] and path planning support based on traveling time analysis [36]), **EDRoad** can also provide an extra layer of data integrity check for mobile CPSes to verify their own status measurements while executing services like collaborative sensing [148] and map construction [35]. For example, a vehicle can ask the infrastructure with **EDRoad** for verification of its own GPS coordinates, or a vehicle with **EDRoad** can act as an "anchor" in a fleet to verify received data from other entities.

This chapter makes the following main contributions:

- Development of **EDRoad**, a new ensemble learning design with expansibility for the verification and recovery of *individual* status data in a multi-data system, including:
  - A new concept of verification tree for achieving easy expansion and enhancing run-time performance (Chapter 4.4); and
  - New run-time data verification and recovery mechanism that will dynamically adapt to the quality of run-time measurements (Chapter 4.4).
- Demonstration of **EDRoad**'s performance as an ensemble learning framework (Chapter 4.5) and as a system approach with $>$95% recall and $<$4% fall-out via extensive evaluation based on real-world trace data (Chapter 4.6).

4.2 Related Work

Prior work has taken two directions to verify the authenticity of a received data or message: (1) by cryptography, such as Message Authentication Code (MAC), or (2) directly

by the data value. Since **EDRoad**'s goal is to verify the data value instead of the integrity of the message sender, we focus on the approaches of verifying data value. While data verification can be considered as anomaly detection in the signal space where an anomaly is defined as the deviation of certain data from its actual value, most prior anomaly detection schemes can be directly applied as data verifiers in their corresponding application scope. Since there are multiple comprehensive surveys summarizing different aspects of (data) anomaly detection in vehicular communications [15, 105, 116, 141] and other CPSes [3, 46, 145, 150], we only mention the prior work directly related to **EDRoad**.

4.2.1 Detection Only

Model-and-compare is the most commonly-used approach in data verification and anomaly detection. Specifically, the verification system will first model the system behavior based on physical/mathematical modeling or machine learning to predict the system state/output and then determine whether the predicted system state matches the received measurements [2, 5, 23, 45, 50, 54, 79, 90, 101, 102, 124, 146, 153, 157]. If not, the verification system will report the detection of an anomaly. Note that since this type of verification system usually models the target CPS system with a single (series of) equation(s), they can only determine whether or not there is an inconsistency in the received data, but cannot identify which data is anomalous. For example, SAVIOR [104] and PID-Piper [28] are the state-of-the-art system-invariant approaches that can be used to construct a vehicle state verification system. However, their design uses a group of data (*e.g.*, vehicle's old location, speed, and acceleration) to predict another (group of) data (*e.g.*, current vehicle location) and, therefore, they can only tell there is an anomalous data used in their formulation without telling which data is anomalous.

4.2.2 Detection and Isolation

Fault Detection and Isolation (FDI) [8, 18, 47, 55, 89, 99, 143, 156] not only focuses on detecting anomalies in the system, but also isolates/identifies the data or measurement that caused the anomaly. FDI usually models the target system as a (differential) equation system and solves the equation system to pinpoint the data in question. Note that because most FDIs are designed for a specific target system (*e.g.*, steering system [156]), their formulations are tied to a fixed set of architecture and parameter settings, thus becoming unable to adapt themselves to other application scenarios without engineers' manual adjustment of the detailed system formulation.

4.2.3 Vehicular Communications (for Case Study)

While there are prior studies (*e.g.*, [117]) focusing on detection of false alerts (*e.g.*, emergency brake and blind spot warning) for connected vehicles by cross-validating the received vehicle states, traffic predictions, and the received alerts, **EDRoad**'s case study focuses on detecting incorrect/inaccurate vehicle states (*e.g.*, location, speed, etc.) received by an RSU. Utilizing wireless measurements, such as received signal strength (RSS), angle of arrival (AoA), and Doppler shift, with multi-modal sensor fusion to perform vehicle state estimation is common for data verification in vehicular communications [1, 13, 118, 126, 127, 129, 147, 151, 152]. These approaches are mainly tailored to work in specific deployment scenarios with specific sensors and (multi-entity) cooperation requirement (Table 4.1) and, therefore, they can be used for data verification in a fixed application settings but cannot be easily adapted to different usage scenarios.

There have also been proposals of using rule-based detection [63, 78] and subjective logic integration [32] to achieve expandable/adaptive detection scope. [63] proposed a Kalman Filter-based detection framework that can also be extended to cover more data types and has features similar to **EDRoad**. While a mix-and-match with prior system approaches (*e.g.*, [1] and [63]) may cover more data types, there is no easy way to combine

individual data verification/recovery and flexible detection scope *simultaneously* without a complete system re-design and overhaul. Because most, if not all, of existing frameworks only propose their system design at a conceptual level without providing any concrete algorithms for actual system implementation/integration, engineers still need to handcraft the algorithms/procedures to provide the required functionalities in addition to detecting inconsistencies among the received data.

4.2.4 Ensemble Learning

Ensemble learning is commonly used to combine multiple (weak) classifiers to construct a stronger classifier that achieves better performance. While anomaly detection systems can be considered as classifiers that output either "normal" or "anomalous", they can be combined with ensemble learning to expand their detection scope. Random forest [59] is an iconic bagging algorithm that combines decision trees (*i.e.*, the weak classifiers) to form a strong classifier. Adaptive boosting (AdaBoost) [42] is another commonly used technique that systematically generates a weight for each of its input classifier during its training process. During run-time, the weight of each classifier is used as the weight of their classification results and the class with the largest weighted sum will be chosen to be the final output. While AdaBoost is originally designed to greedily minimizing the error rate of classification, there are other algorithms designed to enhance AdaBoost w.r.t. different optimization targets and application scenarios with imbalanced training data (*e.g.*, Gentle Adaptive Boosting [43] and Random Undersampling Boosting [119]).

4.2.5 Why EDRoad?

Specifically, **EDRoad** is different from prior work in five perspectives. First, **EDRoad** is designed as a framework to help engineers build a verification system with an emphasis on multi-data verification for identifying *individual* anomalous data instead of constructing a system tailored for a single architecture like most FDIs and detection-only systems. Sec-

Features/Characteristics		**EDRoad**	[147]	[151]	[126]	[1]	[118]	[13]	[152]	[127]	[129]	[78]	[32]	[63]
System or Case Study Properties	Covered Data	p, v, a, h, ω	p	p	p	p	p,v,h	p,h,v	p,a	p,v	p,a	N/A	N/A	p,v
	Required Data	AoA, Map	RSS, LO	RSS, LO	RSS, LO	AoA	Doppler	RADAR, LO	Map	Mobility History	AoA, Doppler	-	-	LO
	Target Scenario	V2I	V2V	V2V	V2V	V2I	V2V	V2V	V2V	V2V	V2V	V2V	V2V	V2V
	Reported Performance (%)	TPR=95 FPR=4	TPR=95 FPR=5	TPR=100 FPR=17.4	TPR=83.7 FPR=3.6	TPR=48.3 FPR=9.2	TPR=99.9 FPR=0.1	-	TPR=90 FPR=2.7	-	TPR=93.5 FPR=3.5	-	-	-
Features (Y is better.)	Individ'l Data Veri. & Recovery	Y	Y	N	N	Y	N	N	N	N	N	N	N	N
	Easy Expansion	Y	N	N	N	N	N	N	N	N	N	Y	Y	Y
	Easy Config.	Y	N	N	N	Y	N	N	N	Y	Y	Y	Y	Y

Table 4.1: Comparison of **EDRoad** and prior work, where $\langle p, v, a, h, \omega \rangle$ are vehicle $\langle$location, speed, acceleration, heading, yaw rate$\rangle$, V2V/I is vehicle-to-vehicle/infrastructure, TPR (FPR) is true (false) positive rate, and LO is short for long-term observation.

ond, **EDRoad** does *not* require engineers to design observers or classifiers for performing anomaly detection, but to only provide the correlation/causality between data (*i.e.*, the domain knowledge). **EDRoad** has mechanisms to automatically transform those correlations into observers/classifiers for detecting anomalous data. Third, **EDRoad** not only detects the occurrences of data anomalies but also restores the thus-identified anomalous data. Fourth, unlike most prior anomaly-detection schemes that do not provide a clear error-tolerance setting and optimization goal, **EDRoad** provides mechanisms for engineers to configure their preference, and maximizes (minimizes) the expected lower (upper) bound of average detection (false-positive) rate under the thus-provided configuration.

Lastly, **EDRoad** can be directly used as an ensemble learning framework if advanced observers/classifiers are already available and shows superiority to prior ensemble learning [43, 58, 59, 66, 119] w.r.t. performance and stability in multi-data anomaly settings (Chapter 4.5). Function-wise, while prior ensemble learning methods for classification can be adapted for individual data verification by assigning each data type a classification label, they neither recover anomalous data nor construct a classifier automatically. That is, even though ensemble learning can be directly utilized to combine existing anomaly detection schemes to expand their data coverage, engineers still need to (manually) design individual detection mechanisms as the input to ensemble learning if s/he needs to include certain data that are not covered by some readily available anomaly detection. Also, they do not provide feedback (during design time) w.r.t. whether the given classifiers can actually perform individual data verification, leading to potentially producing the same label for a certain set of data all the time without informing the user (Chapter 4.4.2.2). To the best of our knowledge, there still does not exist any end-to-end framework to help engineers construct a data verification system that treats different data types as individual elements and utilizing the cyber-physical correlation/causality as additional information to enhance its performance with concrete mathematical property descriptions (Chapter 4.4).

4.3 Problem Formulation and Threat Model

Formulation. Given at least one measurement or an input data with bounded error, **EDRoad**'s goal is to i) determine whether each data is normal ($\mathscr{N}$) or anomalous ($\mathscr{A}$) and ii) restore the data determined to be anomalous. Specifically, a data anomaly is defined as the condition in which a value recorded in a message M_k, denoted as $x_{i,k}$, deviates from its ground-truth value $X_{i,k}$ by more than the error tolerance of that particular data type set by the engineer, where i is the data type index. The error tolerance or detection threshold is Γ_i (Λ_i) if it indicates an absolute (relative) deviation. So, data $x_{i,k}$ is said to be *anomalous* (*i.e.*, $x_{i,k} \in \mathscr{A}$) if

$$|x_{i,k} - X_{i,k}| > \Gamma_i \text{ or } \left|1 - x_{i,k}/X_{i,k}\right| > \Lambda_i; \tag{4.1}$$

otherwise, data $x_{i,k}$ is said to be *normal* (*i.e.*, $x_{i,k} \in \mathscr{N}$). Γ_i's and Λ_i's should be set to values smaller than the requirement of CPS services. If the target service does not have any specific requirement, the thresholds can be set to the maximum noise level or a certain percentile observed in the training data as in our case study (Chapter 4.6).

Case Study. To facilitate a better illustration while introducing our system design, we will use BSM verification (Fig. 4.1) as concrete examples throughout this chapter and it will also be used for our case-study evaluation. Specifically, **EDRoad** is assumed to be deployed in an RSU for the verification of vehicle location (p), speed (v), acceleration (a), yaw rate (ω), and heading (h) in the received BSMs. The RSU is also capable of angle-of-arrival (AoA or ϕ) measurement and have access to the road/map information ($\mathbb{M}$) of the surrounding region.

Threat Model. We assume a *strong* adversary with the full knowledge of **EDRoad**'s design and parameters. The attacker has the goal of tricking a target entity, where **EDRoad** is deployed, to use incorrect data to provide incorrect/inaccurate services to mobile CPSes. S/he can manipulate any data transmitted by external entities (*i.e.*, not ego entities). How-

ever, s/he cannot fully control at least one "trusted" input to **EDRoad** (*i.e.*, this input can only be manipulated to deviate from the ground truth by a certain bound).[2] In practice, a trusted input can be obtained from one of the following sources:

- A sensor with secure hardware or redundancy design, which makes its signal inherently hard to falsify without physical access to the sensor (*e.g.*, AoA [149]);
- A system command/setpoint that only goes through the internal network of the ego system (*i.e.*, where the detection system is deployed), which is the implicit assumption of most prior work (*e.g.*, [55, 104]); or
- A static reference information (*e.g.*, road/map information).

The adversary may simultaneously and continuously manipulate multiple data types to evade **EDRoad**'s detection. We would like to stress that **EDRoad** is designed for data verification, *not* for detecting every possible type of attack. An attack will only be **EDRoad**'s detection target if and only if there is at least one manipulated data that can satisfy the anomalous condition introduced earlier. However, this threat model still includes the "stealthy attacks" (*e.g.*, Drift-with-Devil [120]) as long as they eventually cause some data to deviate from their ground truth by greater than the thresholds.

4.4 System Design

4.4.1 Workflow Overview

Like most detection systems, **EDRoad** consists of two phases (Fig. 4.2): *Training* and *Execution*. During the training phase, **EDRoad** will ask the engineers to provide system descriptions (SDs) that depict the domain-knowledge of the target data. Specifically, each system description should capture a certain correlation/causality between two or more data in the form of:

$$z_\tau = \mathcal{F}\left(x_{1,k}, \ldots, x_{n,k}, x_{1,k'}, \ldots, x_{m,k'}\right), \tag{4.2}$$

[2] This is a requirement for *all* systems that design to identify individual anomalous data.

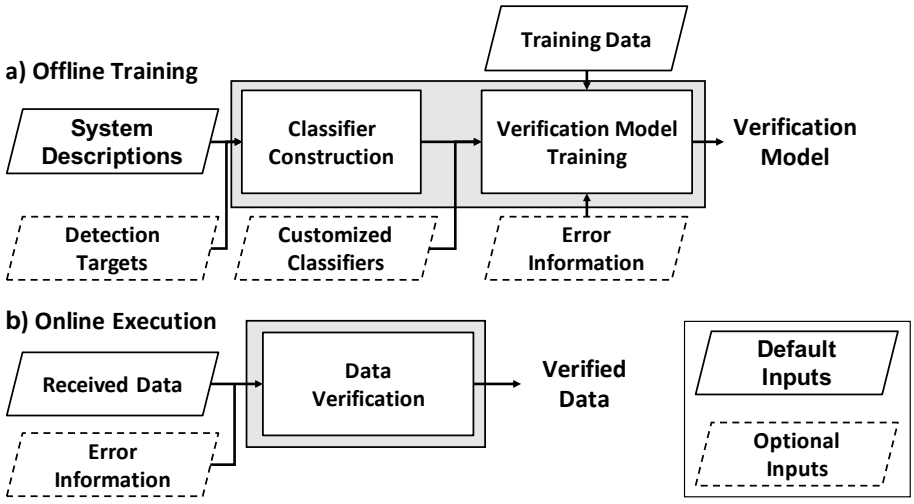


Figure 4.2: System overview of **EDRoad**.

where $x_{i,k/k'}$ is a data with timestamp index k/k' and data-type index i, z_τ is another data with timestamp $\tau = k$ or k', and $\mathcal{F}(\cdot)$ is the function depicting the data correlation.

EDRoad will then take the SDs as inputs and transform them to their corresponding classifiers, each of which checks if the data correlation/causality matches each SD at runtime. These classifiers will first estimate the ground-truth values of their data of interest and report the data to be normal or anomalous according to the criteria defined in Eq. (4.1). Note that engineers can also directly input/supply classifiers to configure **EDRoad** to customize the detection and skip the above step.

Next, **EDRoad** will use the generated classifiers and the training data to establish a verification model. During this step, **EDRoad** will also provide feedback to the engineers w.r.t. (i) whether the provided SDs can meet the requirement for individual data verification and (ii) what types of SD are missing if the requirement cannot be met. Finally, **EDRoad** will verify data based on the model obtained during the training phase. Specifically, **EDRoad** verifies the received data, one at a time, with different combinations of classifiers and further uses the verified/restored data to check another unverified data. **EDRoad** will repeat this process until all received data are verified.

4.4.2 Training Phase

4.4.2.1 Construction of Classifiers

The first step in **EDRoad**'s training (Fig. 4.2a) is to transform the SDs and the error tolerance into classifiers. Specifically, each classifier takes the data types specified in the corresponding SD as inputs and outputs a binary result, *i.e.*, "normal" (0) or "anomalous" (1).

EDRoad utilizes the estimate-then-compare approach to construct classifiers. Each SD input will be transformed into a corresponding estimator which will be used to estimate the ground-truth system state based on the received data. If the estimated state deviates from the received data by a value greater than the error tolerance, this classifier will report the detection of an anomaly. The default true state estimation that we use in **EDRoad** is based on Maximum Likelihood Estimation (MLE) [86]. The choice of MLE is grounded on the fact that performing MLE only requires the error distribution of the input data and it can be utilized for both single-point-of-time and time-series data.

Based on the example that an RSU wants to verify the vehicle's position (p_k) in BSM_k, we use an SD that depicts the correlation between i) the AoA measurement (ϕ_k) and ii) the relative locations of the RSU and the target vehicle (Fig. 4.3) to illustrate the process:

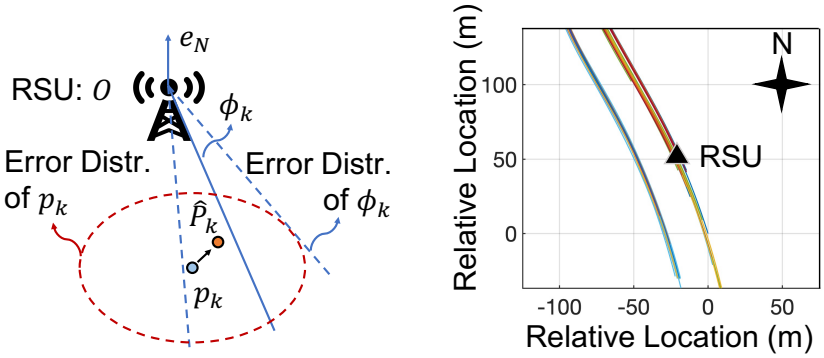
$$\phi_k = \angle e_N O p_k, \tag{4.3}$$

where e_N is the North direction and O is the RSU's location.

The first step of the process is to formulate the likelihood function L based on the probability of obtaining the received data while treating their ground truths as given variables:

$$L(P_k, \Phi_k) = Pr(p_k, \phi_k | P_k, \Phi_k) = Pr(p_k | P_k)\ Pr(\phi_k | \Phi_k), \tag{4.4}$$

where Pr indicates the probability density function (pdf) and P_k (Φ_k) is the ground truth of



(a) True state estimation example. (b) Overview of testing data.

Figure 4.3: RSU's and vehicle's relative locations.

p_k (ϕ_k). Note we use an upper-case letter to represent the ground truth and a lower-case letter to represent the received/measured data. Eq. (4.4) is valid because p and ϕ are measured with different sensors (*i.e.*, GPS in the vehicle and wireless transceiver in the RSU, respectively)[3] and, therefore, their measurement errors can be considered independent. The next step is to substitute the data that is on the left-hand side of the SD (*i.e.*, Eq. (4.3)) using only the data on the right-hand side:

$$\begin{aligned} \text{Eq.}(4.4) &= Pr(p_k|P_k) \times Pr(\phi_k|\angle e_N O P_k) \\ &= Pr(p_k|P_k) \times Pr\left(\phi_k|\tan^{-1}\frac{O_X - P_{k,X}}{O_Y - P_{k,Y}}\right) = L'(P_k), \end{aligned} \tag{4.5}$$

where subscripts X and Y represent coordinates in west–east and south–north directions, respectively. The last step of the estimation process is to solve

$$\hat{P}_k = \arg\max_{P_k} L'(P_k), \tag{4.6}$$

and obtain $\hat{\Phi}_k$ by plugging $\hat{P}_k$ into Eq. (4.3), where $\hat{X}$ denotes X's estimation. The classifier can then compare the estimated values with the received ones to detect inconsistencies, if

[3]Note that the error distribution of p_k can be obtained from the BSM Part I [114] and that of ϕ_k can be obtained from the measurement error estimation.

any, which we call *consistency check.* **EDRoad** may also utilize a Kalman Filter for state estimation if the SD is covering time-series data and the required covariance matrices are available.

Note that **EDRoad** designs classifiers for engineers with little to no knowledge of how to perform a data consistency check. Experienced engineers can design a sophisticated classifier for a given SD, or a classifier may already be available from prior studies. In either case, engineers can use the customized classifier as input to the training of verification model.

4.4.2.2 Training of Verification Model

There are two steps to train the verification model. First, **EDRoad** generates a procedure template for determining the sequence of verifying each input data based on the classifier inputs. Second, **EDRoad** obtains the detail parameters for combining the results of the classifiers via a training process.

1) Generation of Procedure Template. The goal of generating a procedure template is to explore all possible verification orders (*i.e.*, which data to verify first) that **EDRoad** can follow to maximize **EDRoad**'s detection performance. Because the procedure template can be visualized by a tree data structure, we call this procedure template a *verification tree* (VT) template. The VT template is constructed based on a simple rule — a classifier can be utilized to perform a data consistency check if and only if all but one data in that classifier have been verified. We will also use the example of BSM verification (*i.e.*, "Ex1" in Table 4.2) with Fig. 4.4 to illustrate the process of constructing a VT template.

First, **EDRoad** will create a root node and mark the initial trusted data/measurement (ϕ) in this root node (Node-1 in Fig. 4.4a). Each node, say Node-k, in the VT will also maintain a list ℓ_k, recording the data that have been verified so far.

Next, **EDRoad** constructs the VT template node by node based on the classifier (C) candidates that can be used to perform the consistency check in the current state. That

C	p	v	a	h	ω	Δf	ϕ	$\mathbb{M}$	Ex1	Ex2	CS	Formulation (subscripts k and $k+1$ are timestamp indices)
1	•	•							✓	✓	✓	$v_k = \lvert(\vec{p}_{k+1} - \vec{p}_k)\rvert / \Delta t$, where Δt is time interval between k and $k+1$
2	•	•	•	•							✓	$\vec{p}_{k+1} = \vec{p}_k + \int_t^{t+\Delta t} (v_k + a_k \tau)\, \vec{h}_t \, d\tau$
3	•	•	•	•	•						✓	$\vec{p}_{k+1} = \vec{p}_k + \int_t^{t+\Delta t} (v_k + a_k \tau)\, \vec{h}_\tau \, d\tau$, where $\vec{h}_\tau = \vec{h}_k + \vec{\omega}_k \tau$
4	•	•		•						✓	✓	$\vec{p}_{k+1} = \vec{p}_k + \int_t^{t+\Delta t} v_k \, \vec{h}_\tau \, d\tau$
5	•	•		•	•						✓	$\vec{p}_{k+1} = \vec{p}_k + \int_t^{t+\Delta t} v_k \, \vec{h}_\tau \, d\tau$, where $\vec{h}_\tau = \vec{h}_k + \vec{\omega}_k \tau$
6	•	•		•		•				✓	✓	$\Delta f_k = \frac{c}{c - v_{r,k}} f_c$, where $v_{r,k} = v_k \cos(\angle e_N O p_k - \angle h_k)$
7	•			•		•				✓	✓	$\Delta f_k = \frac{c}{c - v_{r,k}} f_c$, where $v_{r,k} = \frac{\lvert(\vec{p}_{k+1} - \vec{p}_t)\rvert}{\Delta t} \cos(\angle e_N O p_k - \angle h_k)$
8	•						•		✓	✓	✓	$\phi_k = \angle e_N O p_k$, see Eq.(4.3)
9	•						•	•	✓	✓	✓	$p_k = intxn(O, \phi_k, \mathbb{M})$
10		•	•								✓	$v_{k+1} = v_k + a_k \Delta t$
11		•		•		•	•			✓	✓	$\Delta f_k = \frac{c}{c - v_{r,k}} f_c$, where $v_{r,k} = \frac{\lvert(\vec{p}_{k+1} - \vec{p}_k)\rvert}{\Delta t} \cos(\phi_k - \angle h_k)$
12		•					•	•	✓	✓	✓	$v_k = \lvert intxn(O, \phi_{k+1}, \mathbb{M}) - intxn(O, \phi_k, \mathbb{M})\rvert / \Delta t$
13				•		•	•	•		✓	✓	$\Delta f_k = \frac{c}{c - v_{r,k}} f_c$, where $v_{r,k} = \frac{\lvert\hat{p}_{k+1} - \hat{p}_k\rvert}{\Delta t} \cos(\phi_k - \angle h_k)$, $\hat{p}_\tau = intxn(O, \phi_\tau, \mathbb{M})$

Table 4.2: An example of classifier (C) or SD data coverage, where Δf, ϕ and $\mathbb{M}$ represent Doppler shift, AoA measurement, and map information, respectively. f_c is the carrier frequency, c is the speed of light, and $intxn(\cdot)$ is the function used to identify the intersection point of the road that the vehicle is traveling on and the virtual line of AoA measurement from the RSU (O). The black dots indicate the data covered in the classifiers, and the check marks indicate the classifiers utilized in the examples (*i.e.*, Ex1, Ex2 and CS). See Chapter-4.4.4 for their detailed descriptions.

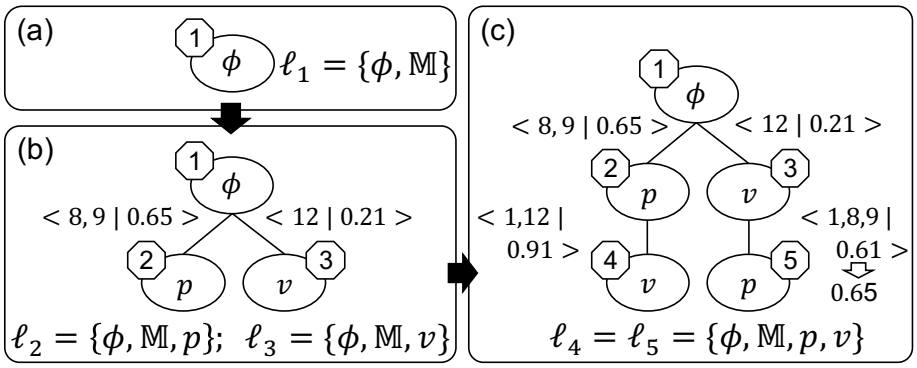


Figure 4.4: An example verification tree (VT), where the numbers on the nodes are the node indices and the tuples on the paths are $\langle$ Classifiers | Youden's Index J $\rangle$. The static information $\mathbb{M}$ can be considered as common trusted information, and hence is not specially marked in the root node.

is, after creating the root node, **EDRoad** will now try to find those candidates that have only one additional data type compared to ℓ_1. The candidates are C_8, C_9, and C_{12}, and their unverified data are p, p and v, respectively. Since there are only two *unique* data (*i.e.*, p and v) can be verified at this point, **EDRoad** adds two child nodes (*i.e.*, Node-2 and Node-3) to the root and marks the corresponding classifiers on the connected paths. The newly-added child nodes will also add the data type marked on them as verified. For example, Node-2 will have $\ell_2 = \{\phi, \mathbb{M}, p\}$ and Node-3 will have $\ell_3 = \{\phi, \mathbb{M}, v\}$ (see Fig. 4.4b). **EDRoad** will repeat the process to all the newly-added nodes until no candidates can be found (Fig. 4.4c). Fig. 4.5 shows another VT example constructed based on "Ex2" of Table 4.2 while assuming Doppler Shift (Δf) is another available and trusted measurement.

Note that the SDs/classifiers provided by the engineers may not be enough to support individual data verification. This condition can be determined by **EDRoad** unable to find a classifier candidate to add a new child node for Node-k, but there are still unverified data. **EDRoad** can then prompt a message to the user on the need of more SDs/classifiers to depict the correlation between ℓ_k and the unverified data.

2) Parameter Training. The goal of parameter training is to obtain the models for combining classifier results and provide information on which verification order **EDRoad** should take at run-time. During run-time, **EDRoad** will verify one data at a time according

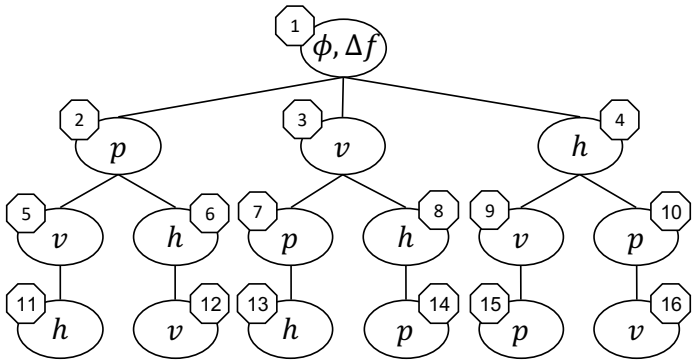


Figure 4.5: VT template of "Ex2" in Table 4.2.

to the VT and this verification procedure is equivalent to finding a path from the root node of the VT to a leaf node. Specifically, **EDRoad** will choose the path that has the best expected performance to traverse down the tree, if possible. Therefore, for each path in VT, the parameter training will obtain i) the models for verifying (M_C) and correcting (M_R) the target data marked in the connected node and 2) its expected performance. The algorithm of run-time verification and its design rationale will be introduced in Chapter 4.4.3.

Other than the classifiers and the VT template, the parameter training has two additional inputs (Fig. 4.2):

– *Training data.* These are (untampered with) data collected during the deployment of **EDRoad** for training the system. They can be regular messages recorded at the deployment site by the engineers themselves.

– *Error Information (Optional).* **EDRoad** can adjust its operation based on the current measurement/data quality. This input lets engineers specify the measurement error levels or distributions that the system should consider during run-time to adjust its operation. For example, if the GPS measurements at the deployment location can sometimes have up to a 10m deviation from the actual vehicle location depending on the weather condition, then engineers may set the GPS to have three measurement levels: $\leq$2.5m, $\leq$5m, and $\leq$10m. Otherwise, engineers can omit this option and use the worst-case error. Note **EDRoad** will create one VT for each error level because the classifiers may exhibit different detection capabilities at run-time, leading to different training results. **EDRoad** will then choose the

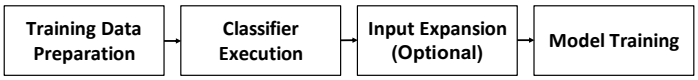


Figure 4.6: Classification model training process.

corresponding VT during its online execution.

Classification Model (M_C) and Expected Performance Training. The goal of M_C training is to obtain M_C for combining detection results of classifiers on a single path in the VT. Fig. 4.6 shows the workflow of M_C training. The main idea of this training is to construct a strong classifier through a modified version of AdaBoost for the detection of data anomalies based on the classifiers. For example, the classifiers to be combined for the path from Node-1 to Node-2 in Fig. 4.4 will be C_8 and C_9. To do so, **EDRoad** needs to prepare a set of training data with anomalies and the ground truth (GT) labels indicating whether a specific training data is anomalous. By default, **EDRoad** will first randomly select data entries from the input training data, shift the data by the value of threshold input to create anomalies, and mark the manipulated data as anomalous. This process simulates the worst-case scenario where attackers know the detection thresholds and manipulate the data right at the threshold values. Experienced engineers can also include any specific scenarios the system should be aware of in the training process.

After generating the training data, the next step is to prepare the necessary inputs to AdaBoost. The first input is the ground truth (GT) labels for each training data, which are already available during the training data preparation. The second input is the matrix of classification results, which can be obtained by directly applying the classifiers to the training dataset. Fig. 4.7a shows an example input of M_C training.

EDRoad then uses the following training process (adapted from the classic AdaBoost) to assign each classifier a weight:

Step 1: Initialize all training data $x_i \in \{0,1\}$ to have an initial weight of $w_i = 1/N_D$, where i is the index of the training data and N_D is the number of training data.

Step 2: Compute the Youden's Index J [155] for each classifier (see the following descrip-

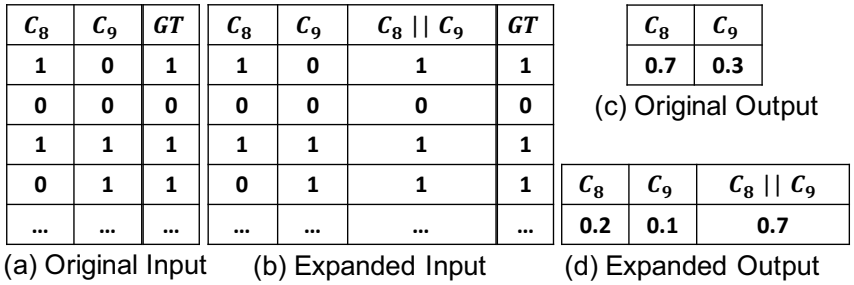


Figure 4.7: I/O examples of M_C training.

tion).

Step 3: Select the classifier C^* with the largest J and assign a weight $\alpha^* = 0.5\ln[(1-\varepsilon^*)/\varepsilon^*]$ to C^*, where $\varepsilon^* = \sum_{\forall C^*(x_i)\neq y_i} w_i$ is the weighted error rate, $y_i \in \{0,1\}$ is the ground truth label of x_i, and $C^*(x_i)$ means applying C^* to data x_i.

Step 4: Update the weighting of training data $w_i \leftarrow w_i exp\left(-\alpha^*(-1)^{y_i+C^*(x_i)}\right)$ and further normalize the sum of data weights to 1, *i.e.*, $w_i \leftarrow w_i/(\sum w_i)$.

Step 5: Repeat Steps 2–4 until all classifiers have been assigned a weight.

The above training process greedily assigns a weight to the classifier with the best performance in each iteration and emphasizes on the incorrectly classified data in the next iteration by increasing their weights. We refer the interested readers to the original derivation of AdaBoost [42] for the rationale and mathematical meaning behind the weight assignment in Steps 3 and 4.

Input Expansion. Since there can be paths in a VT with only two or fewer classifiers, performing AdaBoost on such paths is equivalent to only utilizing the classifier with better expected performance and there will be no performance boosting over use of that particular classifier. For example, if C_8 and C_9 cover different detection aspects, the best way to integrate these two classifiers is to report a data anomaly if any of them detects an anomaly, which is not covered by direct utilization of AdaBoost. Therefore, we introduce an optional step to expand the input matrix before performing the training. The idea is to include not

only the original output from the classifiers but also their combinations. Using the same example, we can include an additional column of $(C_8||C_9)$ as shown in Fig. 4.7b, where "||" is the OR operation.

Youden's Index. Another modification we made to AdaBoost is to change the classifier selection (*i.e.*, Steps 2 and 3) in each of its training iterations from the classifier with the smallest error rate (*i.e.*, the largest correct rate) to that with the largest Youden's Index J [155].

$$J = \text{sensitivity} - \text{specificity} - 1 = \text{TPR} - \text{FPR}, \tag{4.7}$$

where TPR and FPR are true and false positive rates of the weighted classification results, respectively. This modification is made to improve the stability of training results, because the error rate will change according to the ratios of positive and negative samples in the training data even if the classifier has a consistent TPR and FPR performance (*i.e.*, the training results will be highly dependent on the positive sample ratio in the provided training data). On the other hand, J will remain consistent if the classifier has a consistent detection capability. This is also the reason why we choose J to represent the expected performance for M_C, and we will discuss more on the benefits of choosing J over other performance metrics when we introduce **EDRoad**'s online execution in Chapter 4.4.3.

Finally, before assigning the training result $\hat{M}_C$ as the final M_C for a path, **EDRoad** checks whether the grandparent of the connected node already provides a path with a better classification model M'_C to verify the same target data. If not, $\hat{M}_C$ will be used as M_C; **EDRoad** will otherwise use M'_C as M_C. We will henceforth denote Path-i to represent the path from some node to Node-i. For the example in Fig. 4.4c, Path-5 represents the path from Node-3 to Node-5. If **EDRoad** is currently training M_C for Path-5, it will check whether there is any path from Node-1 that verifies the same data as Node-5 (*i.e.*, p) but with better performance. In this example, the existing model from Path-2 has a better

detection performance than Path-5 (*i.e.*, $0.65 > 0.61$) and, therefore, the model of Path-5 will be replaced by the model of Path-2.

This mechanism ensures a data scheduled to be verified later will have the same level or higher expected performance, which will further ensure the detection procedure during runtime is equivalent to greedily maximizing the lower bound of TPR and minimizing the upper bound of FPR (to be proved in Chapter 4.4.3). Note that the model-replacing mechanism will always be valid (*i.e.*, there are sufficient verified data to perform the detection for the current path) because the set of verified data will monotonically increase when traversing down the VT. While **EDRoad** can directly apply M_C to the training dataset to obtain the expected performance based on Youden's Index, we choose to use a two-fold cross validation to avoid over-fitting the training data in computing expected performance on each path.

Data Recovery Model (M_R) Training. Since **EDRoad** performs data recovery based on Gradient Boosting algorithm [58], it follows the standard training process of Gradient Boosting for its M_R Training. The idea of data recovery is to estimate an unverified data based on relevant verified data and **EDRoad** also performs M_R Training for each path in the VT. On each path, the data type to be restored (*i.e.*, the data type marked in the connected node) will be treated as the supervised signal and the other data types covered in the classifiers of the target path will be the regular training input. Take the path from Node-2 to Node-4 in Fig. 4.4c as an example, p, ϕ, and v will be included in the training data and v is specifically treated as the supervised signal. The training result will be a Gradient Boosting model of $M_R : (p, \phi) \rightarrow v$.

4.4.3 Data Verification

4.4.3.1 Runtime Verification

EDRoad can be set to perform data verification periodically or upon receipt of a message containing one or more data types. **EDRoad** takes three inputs to perform data ver-

ification (Alg. 1). They are i) the VTs obtained from the training phase ($\mathscr{V}$), ii) the messages/data that need to be verified, and iii) the (optional) measurement quality estimation (α_v) upon receipt of each measurement/data (Fig. 4.2). Note that the received data/measurement quality estimations are stored in buffers $\mathscr{Q}$ and $\mathscr{E}$, respectively, before processing them.

Since classifiers in **EDRoad** can sometimes involve data from two or more consecutive messages, there may be cases that a data with timestamp k can only be verified when another data with timestamp $k+1$ is received and verified. Take the classifier constructed from the following SD as an example:

$$v_k = |(\vec{p}_{k+1} - \vec{p}_k)|/\Delta t, \tag{4.8}$$

The vehicle speed v_k can only be verified when both vehicle locations p_k and p_{k+1} are verified. Based on this observation, we propose *Forward Creation and Backward Execution* (FCBE) mechanism for performing data verification. Upon receiving a message, **EDRoad** will create a new instance of VT corresponding to the measurement level at that time, which records the verification progress of that particular message (Line 4 in Alg. 1), and then puts it in treeQueue (Line 5). **EDRoad** will then perform verification on each VT in treeQueue in *reverse* direction, meaning that the most recently created VT will be executed before the previously created VT (Lines 8 and 10).

4.4.3.2 Verification Overview in a Single VT

The verification process in single VT (Alg. 2) is akin to finding the best path from the root to a leaf node based on the models and the expected performance specified on each path of the VT. Starting from the root, **EDRoad** will check whether the path with the best expected performance (*i.e.*, having the largest Youden's Index J) is ready to perform data consistency check (Line 5 in Alg. 2). If the path is ready (*i.e.*, all required data for the clas-

Algorithm 1: Runtime Data Verification — FCBE

```
1  Function: data_verification (𝒬, ℰ, 𝒱);
   Input  : Message buffer (𝒬) and measurement error buffer (ℰ)
            VT set (𝒱).
   Output: Results of verification (ℛ).
2  treeQueue = { };
3  while ¬is_empty(𝒬) do
4      tempVT = create_tree(𝒬, 𝒱, ℰ);
5      push(treeQueue,tempVT);
6      for idx from size(treeQueue) - 1 downto 0 do
7          if idx < (size(treeQueue) - 1) then
8              ℛ[idx] = vt_verification(treeQueue[idx], treeQueue[idx].Message,
                 ℛ[idx+1]);
9          else
10             ℛ[idx] = vt_verification(treeQueue[idx], null, null);
11 return ℛ;
```

sifiers marked in the tuple are received), it will perform consistency check with M_C on that path and try to traverse down to the connected node. That is, if no data anomaly is detected by M_C on the target path, **EDRoad** will move on to the connected node (Node-T) and report the data marked in Node-T as normal; otherwise, **EDRoad** will report the data marked in Node-T as anomalous and perform the same procedure to the path with the next largest J. **EDRoad** will repeat this procedure (*i.e.*, try to perform data verification to the path with next largest J) until all the paths fail the consistency check or it encounters a path waiting for further data (Line 6). If all the paths report data anomaly detected, **EDRoad** performs data recovery to the target data on the path with the largest J via Gradient Boosting and moves on to the connected node. Otherwise, **EDRoad** will quit the current verification process of the current message and wait until either necessary data are available or a time-out, a design parameter, is reached. If the timeout is reached, **EDRoad** will remove all the paths still requiring additional data and perform the verification process only on the remaining paths (Line 9).

Algorithm 2: Verification in a Single VT

```
1  Function: vt_verification (𝒯, M_k, M_{k+1});
   Input  : VT instance (𝒯), message to be verified (M_k), and (partially) verified
            next message (C_{k+1}).
   Output: Verification result of M_k (i.e., ℛ_k).
2  waiting = false;
3  while ¬waiting do
4      newNode = 𝒯.currentNode;
5      if largest_J_path_ready(𝒯,M_k,M_{k+1}) then
6          newNode = traverse_down(𝒯,M_k,M_{k+1});
           // Traverse down or switch path until encounter a blockage
              (Sec. 4.4.3.2).
7      else
8          if timeout_reached() then
9              newNode = timeout_traverse_down(𝒯,M_k,M_{k+1});
10     if is_equal(newNode,𝒯.currentNode) then
11         waiting = true;
12     else
13         𝒯.currentNode = newNode;
14 ℛ_k = extract_result(𝒯);
15 return ℛ_k;
```

4.4.3.3 Design Properties

Choice of Youden's Index (J). *J*'s physical meaning is the distance between the receiver operating characteristic (ROC) and the chance level (where TPR = FPR). It is a commonly used metric to represent the system's likelihood to make an informed detection instead of making a random guess. The larger the value, the better detection results. The range of *J* is [-1,1], where 1 indicates perfect detection (TPR=1 and FPR=0) and 0 indicates that the detection is randomly predicting positive results (TPR=FPR) regardless of the input (*i.e.*, a random guess). Note that $J \in [-1,0)$ indicates that the detection produces opposite labels. Therefore, the system should reverse the detection results.

We now introduce the important properties of *J* that are exploited in **EDRoad**'s detection.

Property 4.4.1. *J will not be affected by the positive ratio of the data if the classifier has*

consistent TPR and FPR.

Proof.

$$J = \text{sensitivity} + \text{specificity} - 1 \tag{4.9}$$

$$= TP/(TP+FN) + TN/(TN+FP) - 1 \tag{4.10}$$

$$= TP/(TP+FN) - [1 - TN/(TN+FP)] \tag{4.11}$$

$$= TP/P - FP/N \tag{4.12}$$

$$= \text{TPR} - \text{FPR}, \tag{4.13}$$

where T (F) represents true (false) and P (N) is positive (negative). Since TPR and FPR are computed by only looking P and N samples, respectively, J is independent of the positive ratio of the data. □

That is, unlike other metrics (*e.g.*, F1 score) that stress the positive predictions and can be influenced greatly by the ratio of positive cases, J provides a consistent summary of system performance regardless of the positive sample ratio of the data. Furthermore, it represents the performance bound of the system as stated in the following properties.

Property 4.4.2. *J represents a classifier's TPR lower bound.*

Proof. $\text{TPR} = J + \text{FPR} \geq J$ ($\because \text{FPR} \geq 0$). □

Property 4.4.3. *J represents a classifier's FPR upper bound.*

Proof. $J \leq 1 - \text{FPR}$ $(\because \text{TPR} \leq 1) \Rightarrow \text{FPR} \leq 1 - J$. □

The above properties further lead to **EDRoad**'s design to choose the path with the largest J during the detection phase (Chapter 4.4.3.2). Below we formally state the performance benefits brought by **EDRoad**'s design.

Property 4.4.4. EDRoad *greedily maximizes the potential growth of (average) TPR lower bound by choosing the Path-α with the largest J from Path-1 to Path-K from Node-γ, where K is the total number of paths originating from Node-γ.*

Proof. Based on Property-4.4.2, the sum of TPR lower bounds for verifying the current set of data (*i.e.*, all data that can be verified immediately after Node-γ) after choosing Path-α can be represented as

$$\sum_{i=1}^{K} J_i' = J_\alpha + \sum_{i \neq \alpha} J_i' = \sum_{i=1}^{K} J_i + \sum_{i \neq \alpha} \varepsilon_i, \tag{4.14}$$

where J_i is the Youden's Index of Path-i and J_i' is the Youden's Index for verifying x_i (*i.e.*, the data that can originally be verified by Path-i) after choosing Path-α. ε_i is the difference between J_i and J_i', and $0 \leq \varepsilon_i \leq 1 - J_i$ since our training design ensures $J_i \leq J_i' \leq 1$ (Chapter 4.4.2.2).

Maximizing the potential growth of average TPR lower bound can then be transformed to maximizing the upper bound of Eq. (4.14):

$$\sum_{i=1}^{K} J_i + \sum_{i \neq \alpha} \varepsilon_i \leq \sum_{i=1}^{K} J_i + \sum_{i \neq \alpha} (1 - J_i) = K - 1 + J_\alpha, \tag{4.15}$$

$$\Rightarrow \ \arg\max_{\alpha}(K - 1 + J_\alpha) = \arg\max_{\alpha} J_\alpha. \tag{4.16}$$

Since we have a fixed number of data to be verified, there is no difference in maximizing the sum or the average of a target metric. □

Property 4.4.5. EDRoad *greedily maximizes the potential decrease of (average) FPR upper bound by choosing the Path-α with the largest J from Path-1 to Path-K from Node-γ, where K is the total number of paths originating from Node-γ.*

Proof. This can be proved similarly to Property-4.4.4. □

4.4.3.4 Computation Cost Analysis

The computation cost of **EDRoad** *during run-time* is determined by the number of data covered and the number of verification groups. While assuming **EDRoad** utilizes MLE (with a closed-form solution) for true state estimation and Gradient Boosting for data reconstruction, the worst-case computation cost for a single BSM verification is bounded by $O(N^2+K)$ where N is the number of the data types covered in **EDRoad** and K is the number of SDs.

4.4.4 System Descriptions in Case Study

We introduce the SDs utilized in **EDRoad**'s case study according to the data types involved in each SD. We assume that BSM with index k is received at time t and BSM with index $k+1$ is received at time $t+\Delta t$.

4.4.4.1 Pure Vehicle Dynamics

As shown in Table 4.2, $C_1 - C_5$ and C_{10} form the basic correlation between vehicle state and dynamics. While C_1 describes the correlation between vehicle location and speed as shown in Eq. (4.8), C_{10} describes that between vehicle speed and acceleration:

$$v_{k+1} = v_k + a_k \Delta t. \tag{4.17}$$

On the other hand, C_2 – C_5 capture the correlation between vehicle location and other vehicle state measurements. They follow the basic formulation of:

$$\vec{p}_{k+1} = \vec{p}_k + \int_{t}^{t+\Delta t} (v_k + a_k \tau)\, \vec{h}_\tau \, d\tau, \tag{4.18}$$

where $\vec{h}_\tau = \vec{h}_k + \vec{\omega}_k \tau$. Note that since not all the data in Eq. (4.18) are utilized in C_2 – C_5 while assuming different data availablility, the data not covered in a certain SD will be

replaced by 0 in Eq. (4.18) for that specific SD.

4.4.4.2 AoA (ϕ) and Map Information ($\mathbb{M}$)

C_8 and C_9 describe the correlation between vehicle location and the AoA measurements:

$$C_8 : \phi_k = \angle e_N O p_k; and \tag{4.19}$$

$$C_9 : p_k = intxn(O, \phi_k, \mathbb{M}); \tag{4.20}$$

where e_N is the North direction (see Fig. 4.3a), O (p) is the RSU (vehicle) location, and $intxn(\cdot)$ is the function used to identify the intersection point of the road that the vehicle is traveling on and the virtual line of AoA measurement from the RSU.

Since one can use Eq. (4.20) to estimate the vehicle location, C_{12} combines the correlation described in C_1 and C_9 for vehicle speed verification:

$$C_{12} : \ v_k = |intxn(O, \phi_{k+1}, \mathbb{M}) - intxn(O, \phi_k, \mathbb{M})|/\Delta t. \tag{4.21}$$

4.4.4.3 Doppler Shift (Δf)

C_6, C_7, C_{11}, and C_{13} describe the correlation between vehicle dynamics and Doppler shifts of received BSMs from the RSU side. They follow the formulation of:

$$\Delta f_k = [c/(c - v_{r,k})] f_c, \tag{4.22}$$

where c is the light speed, f_c is the transmission carrier frequency of BSMs, and $v_{r,k}$ is the relative speed between the vehicle and the RSU. Note that while Doppler shift is not treated as a necessary requirement of **EDRoad**, we added these SDs to demonstrate the expandable detection scope of **EDRoad**.

4.5 Evaluation of Verification Frameworks

EDRoad's evaluation is divided into two parts: a generic ensemble learning framework (this section) and a system approach for our case study (Chapter 4.6).

4.5.1 Evaluation Settings

4.5.1.1 Training and Testing Data.

EDRoad (as a generic ensemble learning framework) takes a set of classifiers and training data (*i.e.*, classification results and their ground-truth labels of the input classifiers) as inputs and outputs a data verification model that determines whether individual data is normal or anomalous. The requirement of the ground-truth normality of each data in the classifiers for this evaluation forced us to generate synthetic datasets with different classifier and sample space settings to make the evaluation of **EDRoad**'s performance independent of a specific dataset with fixed properties. **EDRoad** will also be evaluated with real-world datasets in Chapter 4.6.

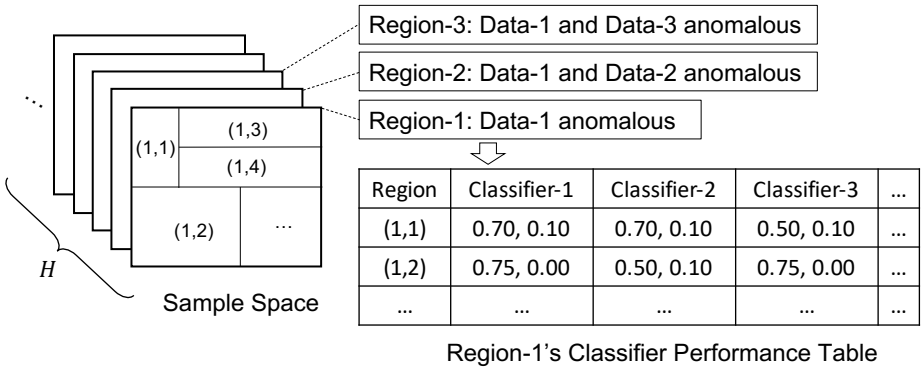


Figure 4.8: A sample space example for data generation.

We assume there are 13 classifiers and 8 data types with 2 trusted inputs as in the case study of BSM verification and the classifiers can only yield moderate detection performance of 0.5–0.75 TPR and 0–0.1 FPR. To generate different detection correlations between classifiers, we first divide the sample space into H (2^{8-2}=2^6=64) regions, each of

which is a possible data anomaly combination, *e.g.*, only Data-1 is anomalous, or Data-1 and Data-2 are anomalous. In each region, there can be up to G (=10) sub-regions of different anomalous scenarios. For each sub-region, classifiers have different correlations with each other. Fig. 4.8 illustrates an example sample space, where we use a tuple (i, j) to represent a sub-region j within region i. For example, Classifier-1 and Classifier-2 have similar TPRs and FPRs in (1,1), *i.e.*, they are likely to have similar detection results, while only Classifier-1 has a high TPR in (1,2). To create the training and testing datasets, we first generate an array of ground-truths with ρ positive ratio and randomly assign a ground-truth sample to each of (region, sub-region) combinations to generate i) the ground-truth labels of individual data and ii) the results of each classifier based on its TPRs and FPRs specified in the performance table of that region (Figs. 4.8 and 4.9).

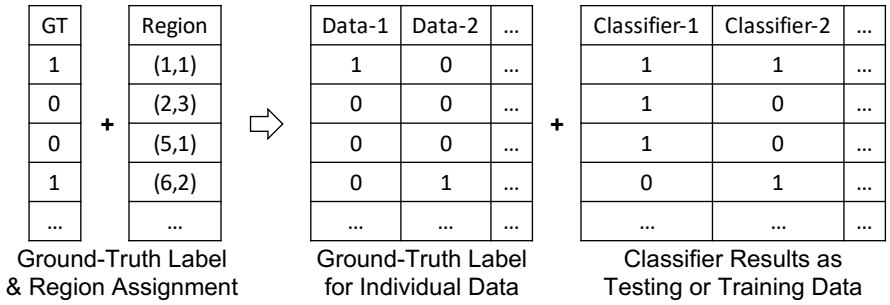


Figure 4.9: An example of training and testing data generation.

4.5.1.2 Baseline Comparison.

We compare **EDRoad** with the most commonly-used or state-of-the-art ensemble learning, including AdaBoost [58], Random Forest [59], Gentle Adaptive Boosting [43], Logistic Boosting [43], Robust Boosting [66], and Random Undersampling Boosting (RUSBoost) [119]. They will be compared using the following metrics.

– *True Positive Rate* (TPR): The probability (per data type) that **EDRoad** successfully detects the occurrence of data anomaly, also known as *detection rate* or *recall*.

– *False Positive Rate* (FPR): The probability (per data type) that **EDRoad** incorrectly

reports occurrence of a data anomaly while it is not, also known as *fall-out*.

– *Youden's Index (J)*: See Chapter 4.4.3.3.

Note other metrics can be converted from TPR, FPR, and the positive sample ratio (ρ) of the data, *e.g.* F1 = (2ρTPR)/[2ρTPR + (1-ρ)FPR+ρ(1-TPR)]. So, they are not explicitly shown here. Also, while **EDRoad** has a more strict applicable scenarios than general-purpose ensemble learning (Chapter 4.4), we only evaluate the test-cases in which **EDRoad** is able to verify individual data.

4.5.2 Results

4.5.2.1 Detection Performance

We first evaluate **EDRoad**'s detection performance for an ideal scenario where the training dataset has the same positive sample ratio $\rho = 0.01$ as the testing dataset, both with 10,000 sample sets. Fig. 4.10 shows **EDRoad**'s performance in comparison with existing ensemble learning frameworks. **EDRoad** is shown to outperform existing frameworks in most testing scenarios by up to 0.51 absolute TPR and 0.48 absolute Youden's index. Even though RUSBoost shows better TPR and J when there is only one anomalous data, **EDRoad** achieves better overall performance when there are more than one anomalous data.

Other $\langle$sample space, TPR, FPR, $\rho\rangle$ settings show a similar pattern as in Fig. 4.10. Figs. 4.12 and 4.13 compare the performance of **EDRoad** with other ensemble learning frameworks under the condition in which the input classifiers have 0.8–1.0 TPR. Figs. 4.14 and 4.15 compare their performances under the condition in which the input classifiers have 0.6–0.8 TPR. Each testing scenarios have 10 different sample space test-cases and each test-case has 10,000 sample sets of training and testing data. We can observe that they all show similar patterns as Fig. 4.10, where RUSBoost has better performance under single-data anomalies while **EDRoad** achieves better performance under multi-data anomalies.

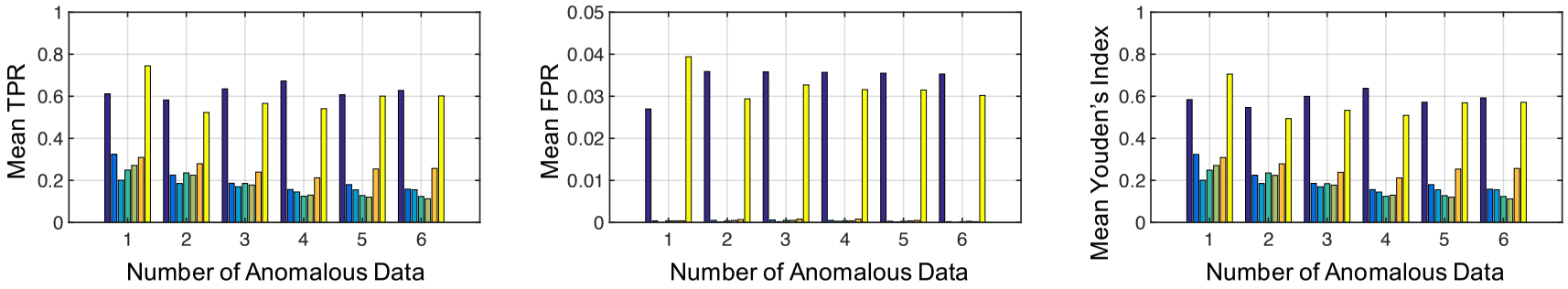


Figure 4.10: Performance comparison of **EDRoad** and ensemble learning frameworks. This figure shares the same legend as Fig. 4.11.

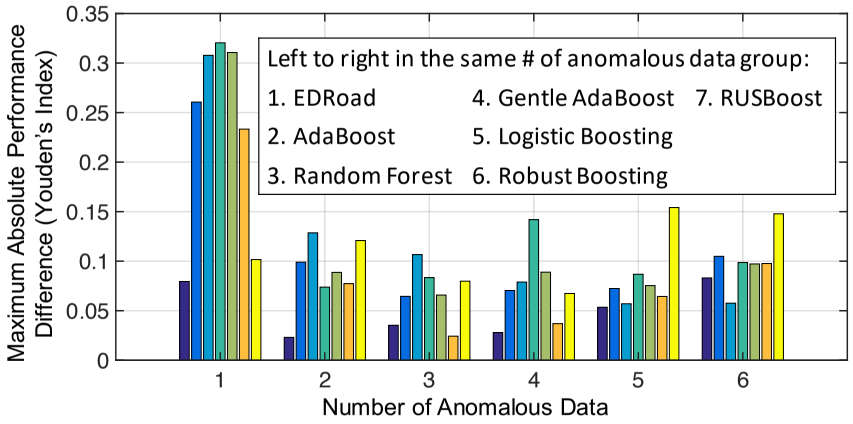


Figure 4.11: The absolute performance deviation caused by inconsistent training and testing datasets.

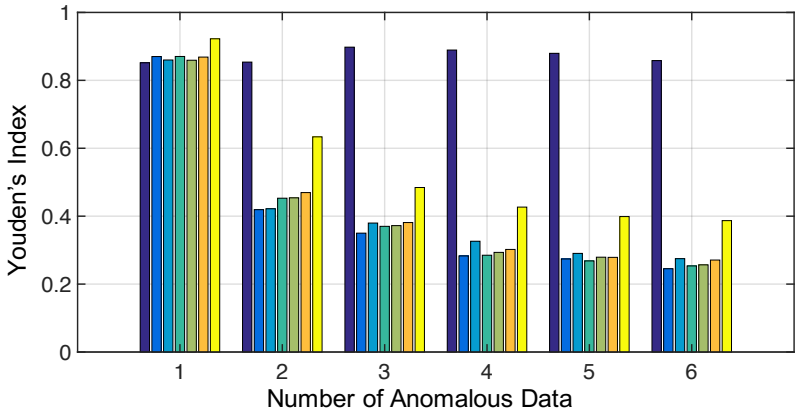


Figure 4.12: Performance comparison when input classifiers have 0.8–1.0 TPR and 0–0.1 FPR with training $\rho = 0.05$ and testing $\rho = 0.01$. The values in each anomaly scenario from left to right are 1) AdaBoost, 2) Random Forest, 3) Gentle Adaptive Boosting, 4) Logistic Boosting, 5) Robust Boosting, and 6) RUSBoost.

4.5.2.2 Performance Persistency

Next, we demonstrate the performance deviations of **EDRoad** and existing frameworks when the training and testing datasets do not exhibit the same characteristics. We fixed the testing data to have 10,000 samples with positive rate $\rho = 0.01$ while adjusting each set of training data to have 10,000 samples with $\rho = 0.005$–0.1 (*i.e.*, 0.5x–10x of that of the testing data). We then compare the testing performance of **EDRoad** and prior approaches with their corresponding performance when the training and testing data have the same positive rate (*i.e.*, $\rho = 0.01$). Finally, we show the maximum absolute performance deviation in different anomaly scenarios (Fig. 4.11).

Ideally, the performance deviation should be close to 0, meaning that an algorithm is unaffected by the dissimilarity/inconsistency between the training and testing data. Fig. 4.11 shows that **EDRoad**'s performance does not change much even if the testing data do not exhibit similar statistical characteristics as the training data. Therefore, using **EDRoad**, engineers do not have to fine-tune the training data to match the statistics of the actual online execution to have good performance.

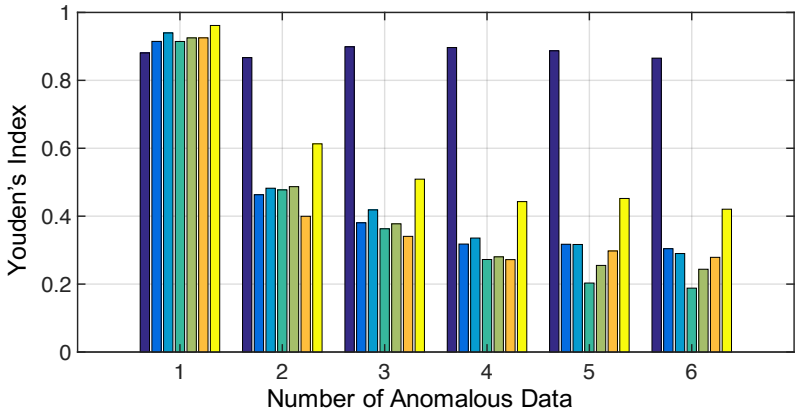


Figure 4.13: Performance comparison when input classifiers have 0.8–1.0 TPR and 0–0.1 FPR with training $\rho = 0.10$ and testing $\rho = 0.01$. This figure shares the same legend as Fig. 4.12.

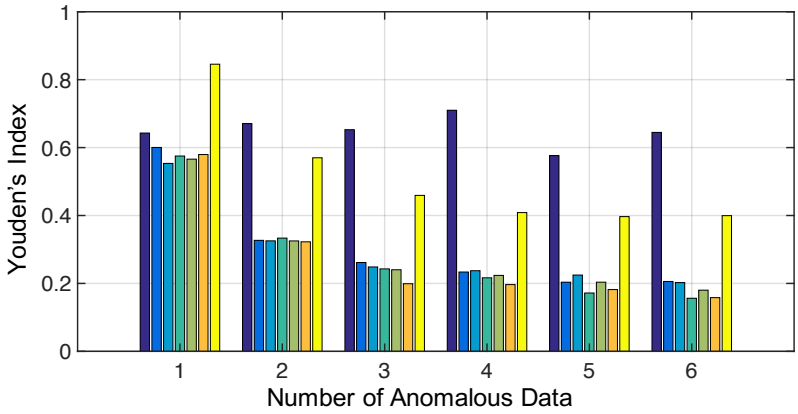


Figure 4.14: Performance comparison when input classifiers have 0.6–0.8 TPR and 0–0.1 FPR with training $\rho = 0.05$ and testing $\rho = 0.01$. This figure shares the same legend as Fig. 4.12.

4.5.2.3 Remark

Unlike prior ensemble learning that determines the final model directly based on classification performance when classifiers are applied to the training data, **EDRoad** performs its training *and* detection according to i) the data types included in the classifiers and ii) classifier performance, making its online execution partially sequential instead of purely parallel to avoid utilizing anomalous data for the verification of another data. While **EDRoad** shows its superiority to existing frameworks for the verification of individual data, it does not always outperform existing frameworks in all possible scenarios. For example, RUSBoost

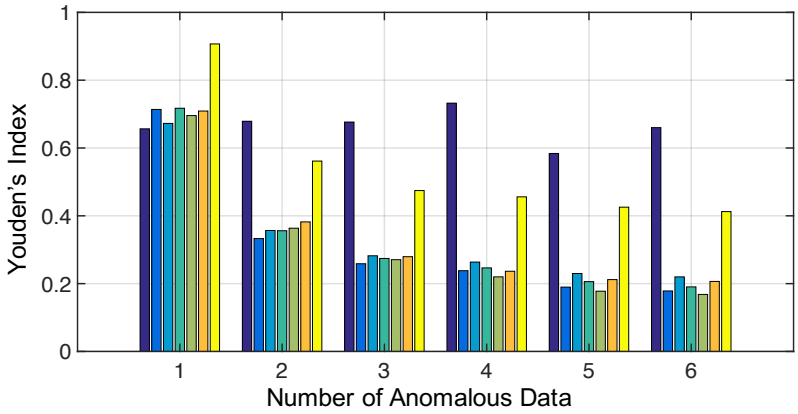


Figure 4.15: Performance comparison when input classifiers have 0.6–0.8 TPR and 0–0.1 FPR with training $\rho = 0.10$ and testing $\rho = 0.01$. This figure shares the same legend as Fig. 4.12.

will provide a better FPR performance than **EDRoad** in determining whether there is a data anomaly while treating all received data as a single group. This is because if **EDRoad** directly reports an anomaly when any of the (individual) data is determined to be anomalous, the thus-computed FPR is equivalent to representing the union of the false-positive conditions of all data, instead of representing the common/intersecting false-positive conditions as in a single-group setting.

4.6 Case-Study Evaluation

4.6.1 Experimental Settings

4.6.1.1 Data Preparation

As described in Chapter 4.3, we assume some malicious vehicles will transmit BSMs with incorrect information to the RSU (*i.e.*, where **EDRoad** is deployed) to prevent the RSU from capturing the correct status of malicious vehicles. In this case study, **EDRoad** verifies the data in BSMs transmitted by the vehicles within the RSU's reception range based on "CS" classifiers/SDs listed in Table 4.2 while assuming AoA and road/map information are the only initial trusted data. To render our evaluation representative of the real-world

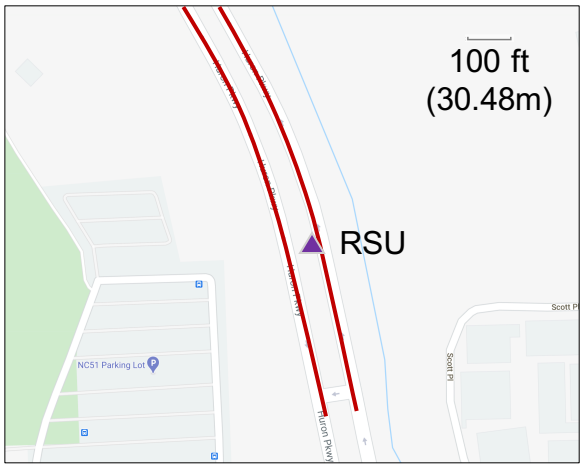


Figure 4.16: Testing data on a map.

driving, we extracted 52 vehicle trips through the same urban road segment (of ∼190m) with a total of 4336 BSMs (10Hz) from Safety Pilot Open Dataset [37]. We use 30 trips (2439 BSMs) as the training data and the rest as the testing data. Fig. 4.3b shows the relative location of the RSU and vehicle traces.

Next, we add Gaussian noises with standard deviation α_m to the testing traces (to control/simulate different levels of natural measurement errors), and manipulate the data on top of the traces with measurement errors to simulate attacks.

Attack Formulation. Adapted from the most powerful GPS/location spoofing reported recently [120] that targets the detection schemes based on Multi-Sensor Fusion (*i.e.*, those similar to **EDRoad**), we implemented an attack targeting vehicle location where all the other data in the BSMs match the manipulated location data. Specifically, the attack implemented in this paper is determined by solving the following optimization problem for $\{(\mathbf{x}_k^a, \mathbf{z}_k^a) | k = 1, \cdots, n\}$:

$$\text{argmin}_{\{\mathbf{z}_k^a | k=1,\cdots,n\}} \Sigma\, D(\mathbf{z}_k^a, \mathbf{z}_k)^2, \tag{4.23}$$

$$\text{where } d_k = D(\mathbf{x}_k^a, \mathbf{P}_k) \text{ and } \mathbf{x}_k^a = M(\mathbf{x}_{k-1}^a, \mathbf{P}_k + d_k, \mathbf{z}_k^a); \tag{4.24}$$

where the subscript k is the timestamp index, the superscript a indicates the manipulated

data during the attack, $\mathbf{x}^a$ is the manipulated vehicle location, d_k is the target deviation at the time k, $\mathbf{P}$ is the ground truth of vehicle location and $\mathbf{z}$ represents other data in BSMs. $D(\cdot)$ is the deviation between the ground truth and manipulated data, $M(\cdot)$ is the process of a Kalman Filter to estimate the next location based on previous estimated location ($\mathbf{x}^a_{k-1}$), current location measurement ($\mathbf{P}_k + d_k$), and other dynamics data in the received BSMs ($\mathbf{z}^a_k$), and the manipulated data should maintain the same noise level as if there were no attacks. The attack starts at a random time after each vehicle enters the communication range of the RSU and lasts for 5s. 5s is long enough to evaluate the time **EDRoad** needs to detect the attack although **EDRoad** actually requires less time as we will show later.

The attack is optimal in the sense that the manipulated data in BSMs will perfectly match each other before noise injection. Also, the attack will *not* introduce any additional, unnecessary high frequency noise to the data. That is, this attack does not leave any tampering evidence for the detection system to perform a side-channel detection by checking if there is a sudden change in the noise level. The (falsified) locations will have drifts of up to 7m (*i.e.*, $\approx$2X the lane width in the US/European countries) from the actual location.

Baseline Comparison. We implemented [63] for comparison due to its most similar design goal as **EDRoad** (*i.e.*, potential to expand the detection scope) and, to the best of our knowledge, it has the most practical deployment design that requires no inter-entity resource sharing for basic deployment. While the basic design of [63] only covers BSM data without incorporating wireless measurements, we expand its detection to further utilize AoA. The basic system proposed in [63] is denoted as **KFV** and the one with AoA support as **KFV/AoA** (*i.e.*, the adapted version of [63] and [1]). In addition to TPR, FPR and J (all computed *per BSM*, not the entire attack duration or a vehicle trip), we also include *Detection Latency* (DL), *i.e.*, the average amount of time required for **EDRoad** to detect an attack, to evaluate **EDRoad**'s performance.

4.6.2 Detection Performance

4.6.2.1 Basic Performance (7m Deviation)

Fig. 4.17 shows the comparison of **EDRoad** with KFV, and KFV/AoA while controlling the location data to have varying (natural) measurement noises of α_m =0.5, 1, or 2.5m. During run-time, **EDRoad** will automatically adjust its detection threshold to 2×**EDRoad**'s estimated measurement error (α_v). From the low TPR performance (<7%) of KFV, we can confirm that the sophisticated location manipulation can indeed evade KFV's detection (*i.e.*, a scheme based on Kalman Filter (KF) without any trusted measurement), corroborating the results shown in prior attack examples [120]. **EDRoad** and KFV/AoA, on the other hand, are able to detect such an attack thanks to their utilization of AoA measurements.

At a first glance, KFV/AoA may seem to have a better TPR performance (86.87%) than **EDRoad** (55.9% TPR) when $\alpha_m = 2.5$, but this TPR performance accompanies the ultimate price of a significantly higher FPR than **EDRoad**. Specifically, while **EDRoad** achieves ≤4.45% FPR in all three scenarios, FPR of KFV/AoA can rise to 53.96% when $\alpha_m = 2.5$. That is, **EDRoad** always has a performance advantage over KFV/AoA in F1 score (with up to 0.33 more F1) as long as less than 46% of data are anomalous. As a result, **EDRoad** outperforms KFV/AoA by ≥18.55% absolute Youden's Index in all three scenarios. Note that when **EDRoad** utilizes $2\alpha_v$ ($= 2\alpha_m$) as the detection threshold, the theoretical expected FPR is 4.5% according to the property of a normal distribution, indicating that **EDRoad** does not incur any significant FPR other than those caused by natural measurement noise. Also, since **EDRoad** will recover/restore the data determined to be anomalous (as we will show later), there is no real harm when **EDRoad** has false-positives. **EDRoad** outperforms KFV/AoA because the latter requires a certain amount of consecutive untampered data prior to the arrival of anomalous data for correct estimation of vehicle state, which cannot be met in this V2I scenario when a sophisticated attacker can start data manipulation right after the establishment of a V2I communication.

Next, we look at the detection latency (DL) of the three schemes (Fig. 4.17b). **EDRoad** is shown to achieve 0.04, 0.07 and 0.19s DL when α_m =0.5, 1 and 2.5, respectively, indicating **EDRoad**'s ability to capture the data manipulation with only 1 BSM in the first two scenarios and with 2 BSMs in the last scenario. According to [19], the planning cycle for signal control systems is typically limited to 5–7s. Since 95% of real-world systems only use 1.2s for planning, **EDRoad** will have sufficient time to detect anomalies. We show **EDRoad**'s detection performance of other data types in Table 4.3.

4.6.2.2 Performance of Varying Attack Magnitudes

We further investigate the performance when a data manipulation attack is mounted by gradually drifting away up to 7m from the vehicle's actual location. Specifically, Fig. 4.18 shows the performance snapshots where the drifting reaches i) less than one lane (1.4m), ii) one lane (3.5m) and iii) two lanes (7m). **EDRoad** outperforms both KFV and KFV/AoA in all three drift magnitudes by >15% absolute Youden's index while maintaining <5% FPR. When we examine the detection latency, **EDRoad** has no problem in detecting a sophisticated location manipulation within 0.8s even with the attack magnitude as low as 1.4m. Although KFV and KFV/AoA seem to have smaller detection latencies than **EDRoad**, this comes from their tendency to generate (false-)positive detections. Specifically, they can only achieve close to, or less than zero Youden's Index, indicating their inability to produce meaningful detection results when the attack magnitude is only 1.4m and >50% of their positive detections may be false-positives.

Next, suppose the attackers can also launch a side-channel attack to influence the error estimations (α_v) of **EDRoad** (*e.g.*, by wireless signal jamming or injecting additional noises into data). Fig. 4.19 shows the change of detection performance when **EDRoad** overestimates the errors embedded in the data by 100 and 400%. **EDRoad** is shown to have less than 6% of performance change regardless of the attack magnitude, and there is only a <0.05s difference in DL when the attack magnitude is 7m. These results demonstrate

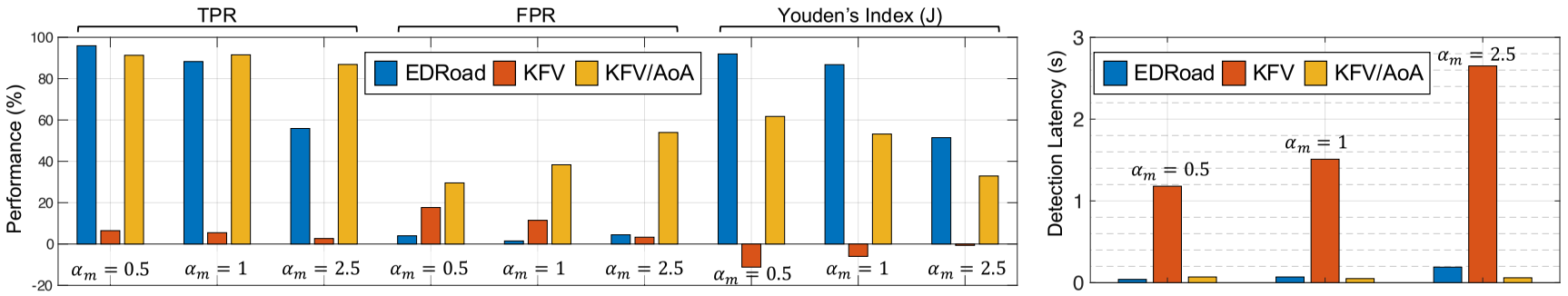


Figure 4.17: Performance and DL comparison with different measurement accuracies and 7m location deviation.

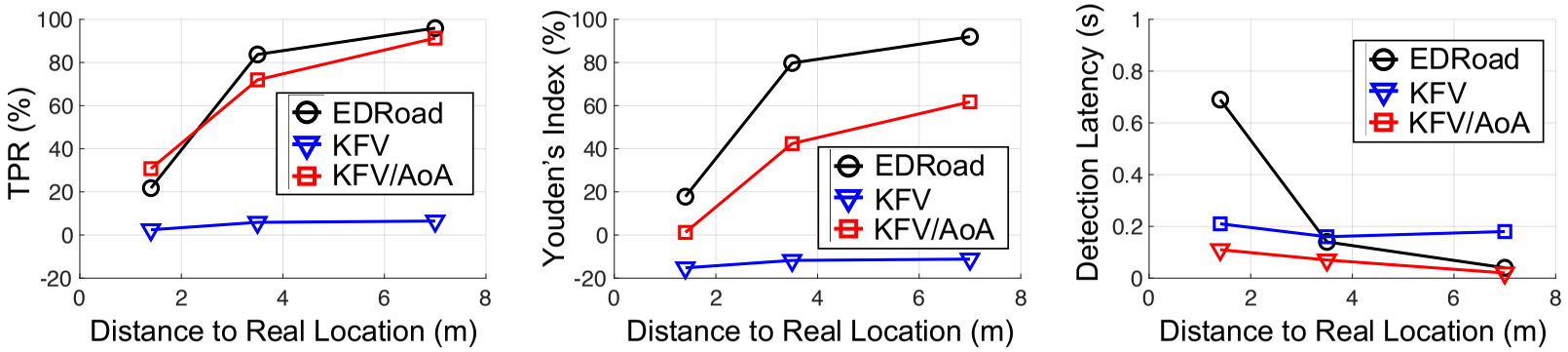


Figure 4.18: Performance comparison with changing attack magnitude. FPR values can be directly derived based on TPR and Youden's Index, so we omit them in this figure.

EDRoad's resilience against i) attackers who try to tamper/interfere with the measurement in the infrastructure and ii) cases when accurate α_v is not obtainable/available.

4.6.2.3 Resilience against Missing Data

Fig. 4.20 showcases the scenario where the speed data are missing/corrupted, or may be filtered out by an outlier test directly (due to an obvious value gap in time domain). **EDRoad** exhibits no observable performance degradation since it inherently considers data in smaller groups and **EDRoad**'s run-time design ignores the unavailable paths in the VT. On the other hand, KFV/AoA's FPR increases drastically, leading to a 30% (absolute value) decrease in Youden's Index.

This large performance degradation is caused by the fact that KFV/AoA is forced to detect anomalies based on noisy acceleration and location measurements for state estimation in the prediction phase of the Kalman Filter (KF) while the update phase is less effective because of the missing data. Note that the results shown here do *not* indicate our state estimation is better than KF (which is known to be optimal with Gaussian measurement noise) in a general setting. Rather, since KF is *not* originally designed for data verification but for state estimation given correct (but noisy) measurements, it is natural for a sophisticated attacker to evade its detection.

4.6.3 Performance of Data Recovery

We use the settings in Table 4.3 and $\alpha_m = 0.5$m for GPS noise level to evaluate **EDRoad**'s data recovery. 1183 BSMs are used for M_R model training and another 1183 BSMs for testing. Fig. 4.21 shows the CDFs of errors in restoring vehicle location and speed. The yellow (leftmost) curve shows the theoretical performance bound computed based on a KF with complete/correct prior measurement and measurement-error standard deviation. The purple (rightmost) curve shows the performance with a KF under sophisticated/strong attacks as shown in the previous evaluation. **EDRoad** is shown to significantly improve the

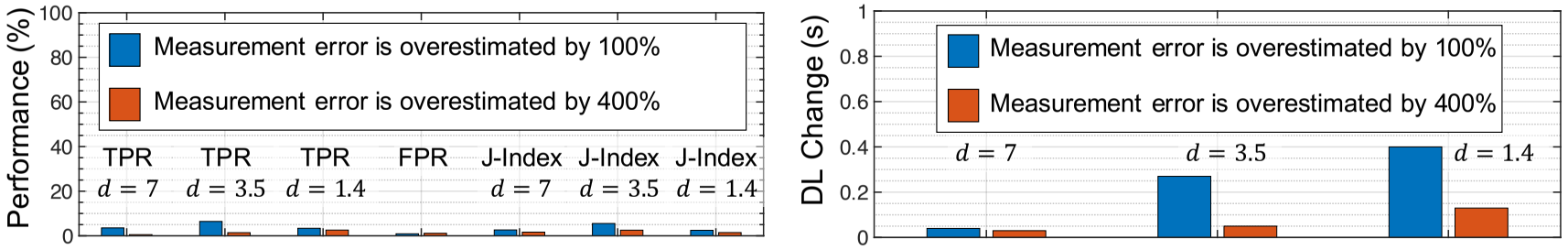


Figure 4.19: Performance and Detection Latency change with incorrect α_v, where d is the attack magnitude.

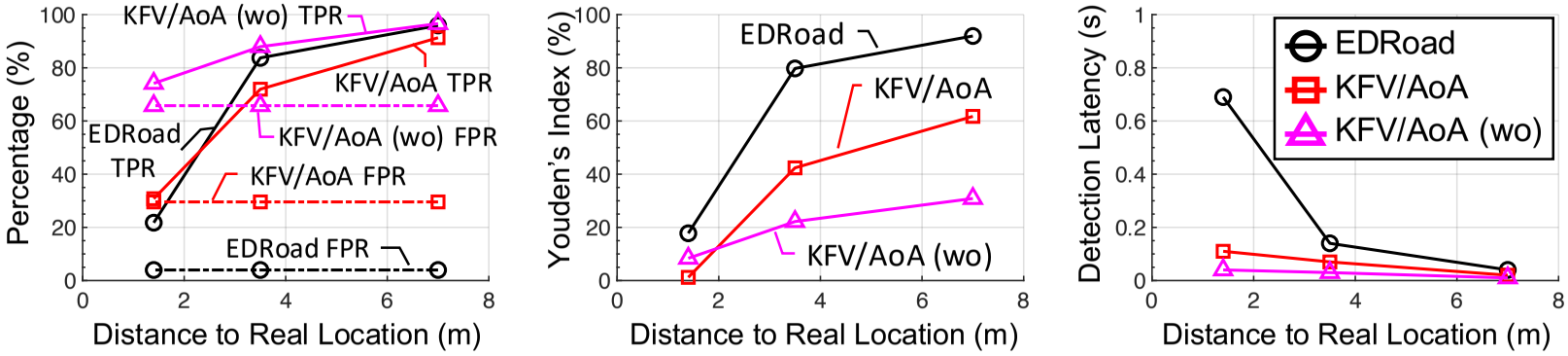


Figure 4.20: Performance comparison with missing data, where "wo" indicates the cases with missing speed data and **EDRoad** does not experience any observable change caused by missing data.

data recovery performance over a KF-based estimator under the sophisticated attacks while maintaining performance close to the theoretical bound.

Data	Unit	α_m	Γ	Δ	TPR	FPR	J	DL(s)
v	kph	1	5	10	100	0.63	99.37	<0.1
a	m/s^2	0.1	1	2	97.64	1.14	96.50	<0.1
h	°	1	10	90	100	13.07	86.93	<0.1
ω	rad/s	0.05	0.08	0.078	100	1.64	98.36	<0.1

Table 4.3: Detection performance (presented in %) of **EDRoad**, where Δ indicates the maximum manipulation level.

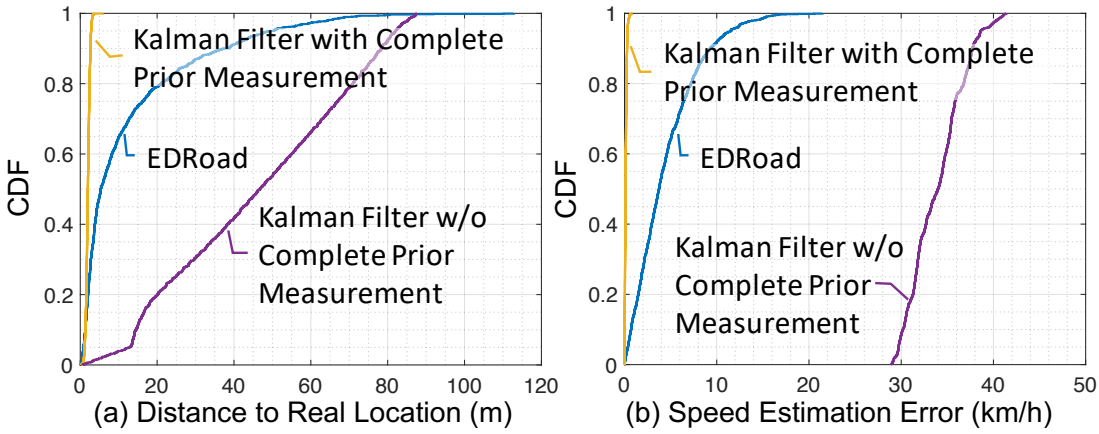


Figure 4.21: Performance of data restoration.

Table 4.4 compares **EDRoad**'s performance of single and full-scale data reconstruction. In the case of full-scale attack, we assume the adversary has the goal to prevent the RSU from getting the accurate vehicle status. Specifically, s/he manipulates every data to deviate from their ground-truths by magnitudes larger than their corresponding thresholds without trying to hide the data manipulation to purposely force **EDRoad** to execute Gradient boosting for data recovery, hoping that the restored data will have less accuracy. Even under such full-scale attacks, the increase of deviation in data recovery is only marginal or even negligible (*e.g.*, the error of restored v increased only 1.5% compared to the ground truth). In summary, it is possible that an attack can trigger a full-scale data recovery, but will only have limited effect on **EDRoad** thanks to its flexible run-time operation. Fur-

thermore, there are no practical benefits of doing so if the attacker is trying to stealthily manipulate the data to further influence the operation of ITS (*i.e.*, avoid **EDRoad**'s detection) since triggering the data recovery also signals **EDRoad**'s detection of an anomaly, which defeats the attacker's purpose.

Data	Unit	Median		Mean		Difference	
		Single	Full	Single	Full	Median	Mean
p	m	5.34	N/A	13.45	N/A	N/A	N/A
v	kph	3.88	5.17	4.59	6.45	0.71	1.28
a	m/s^2	0.06	0.06	0.08	0.08	<0.001	<0.001
h	°	0.07	0.07	0.08	0.08	<0.001	<0.001
ω	rad/s	0.0218	0.0245	0.0276	0.0305	0.0058	0.006

Table 4.4: Deviation between GT and restored data. "Difference" presents the deviation increment from "Single"-data attacks to "Full"-scale attacks.

4.6.4 Supplementary Evaluation Results

Figs. 4.22 – 4.24 show the detailed evaluation results on **EDRoad**'s detection performance in capturing location manipulation (with and without speed data). Specifically, the rows in each figure indicate the actual level of measurement error (α_m) in BSMs while the columns are the potential measurement quality estimation (α_v) obtained from the RSU's wireless transceiver. The inner table of each cell specifies the performance under different <threshold, attack> scenarios. Note that some of the scenarios have an attack level smaller than the threshold and they are not considered as a valid attack. However, we still show their results for an illustration purpose.

Similarly, Figs. 4.25 – 4.27 show the detailed evaluation results on KFV's detection performance. Figs. 4.28 – 4.30 show the detailed evaluation results on KFV/AoA's detection performance. Finally, Figs. 4.31 – 4.33 show the detailed evaluation results on KFV/AoA's detection performance while speed data is missing from the input.

Figure 4.22: **EDRoad**'s TPRs of detecting location data manipulation, where the TPR values are presented in percentage.

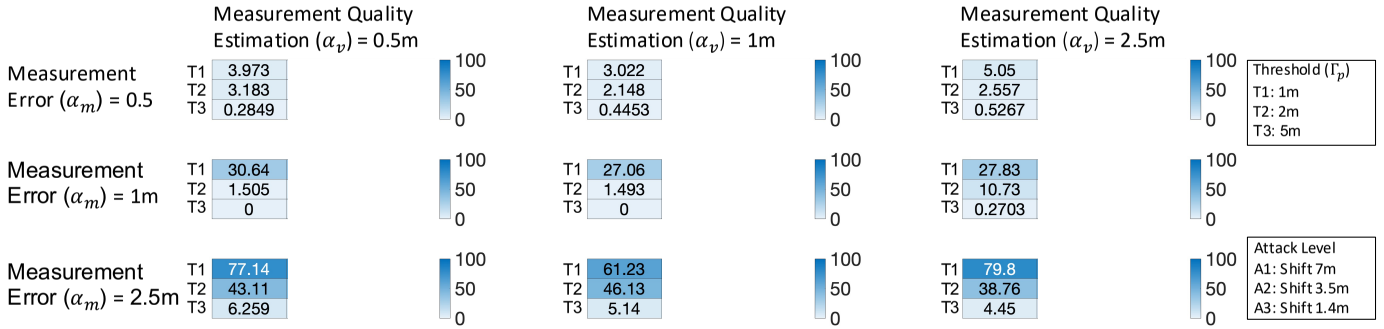


Figure 4.23: **EDRoad**'s FPRs of detecting location data manipulation, where the FPR values are presented in %.

Figure 4.24: **EDRoad**'s detection latency of detecting location data manipulation, where the detection latency are presented in seconds with 0.01s resolution.

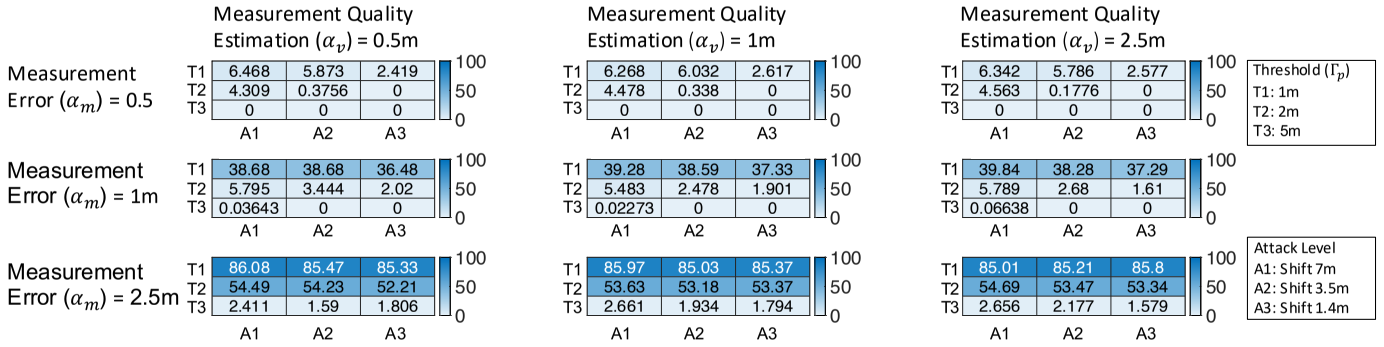


Figure 4.25: KFV's TPRs of detecting location data manipulation, where the TPR values are presented in percentage.

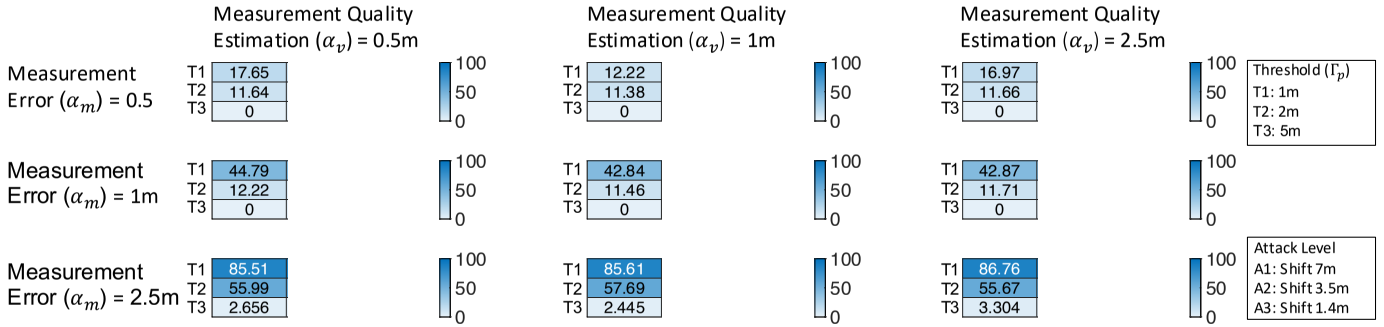


Figure 4.26: KFV's FPRs of detecting location data manipulation, where the FPR values are presented in %.

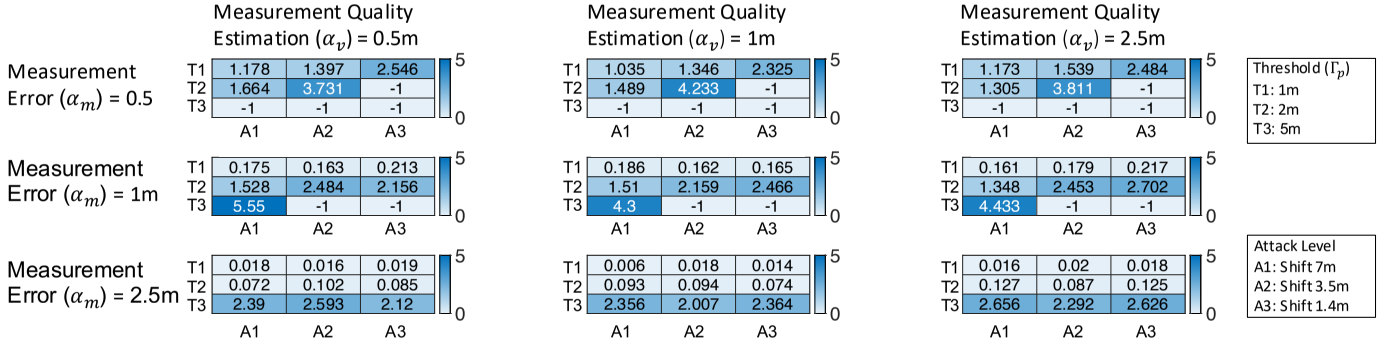


Figure 4.27: KFV's detection latency of detecting location data manipulation, where the DLs are presented in seconds and -1 indicates the attacks cannot be detected.

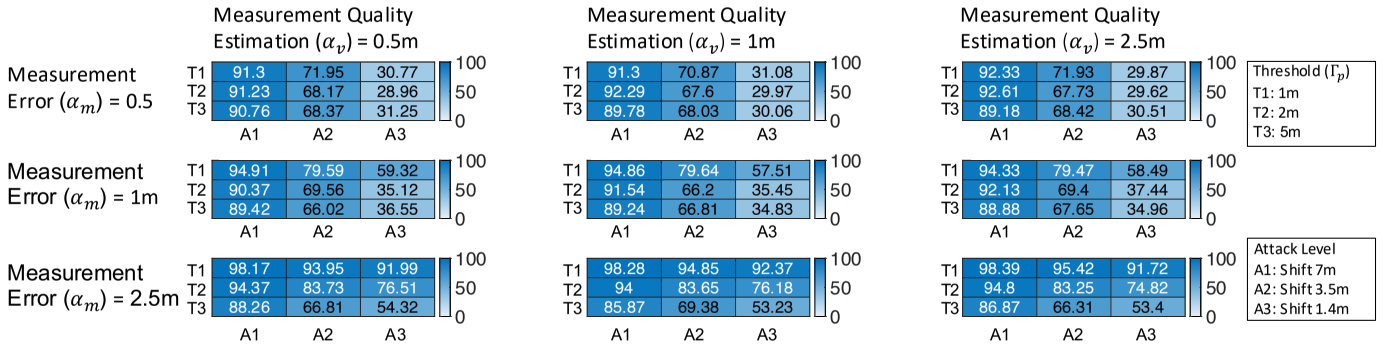


Figure 4.28: KFV/AoA's TPRs of detecting location data manipulation, where the TPR values are presented in percentage.

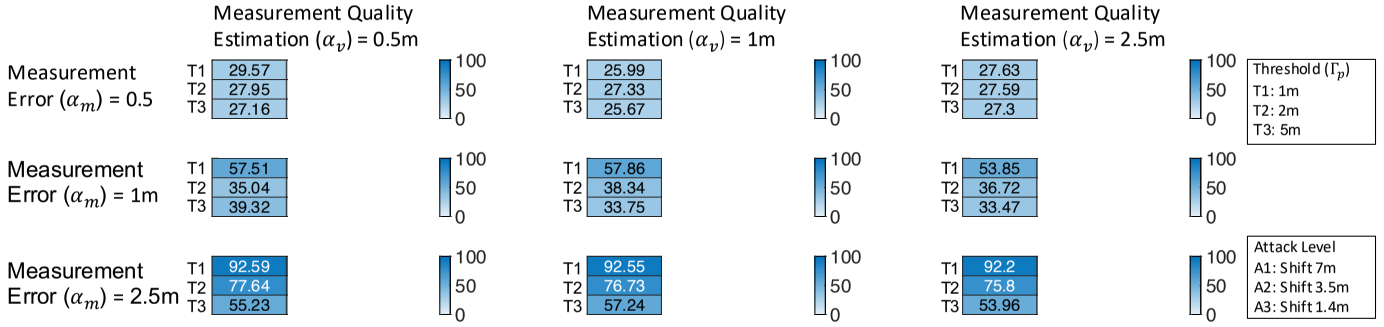


Figure 4.29: KFV/AoA's FPRs of detecting location data manipulation, where the FPR values are presented in %.

Figure 4.30: KFV/AoA's detection latency of detecting location data manipulation, where the DLs are presented in seconds.

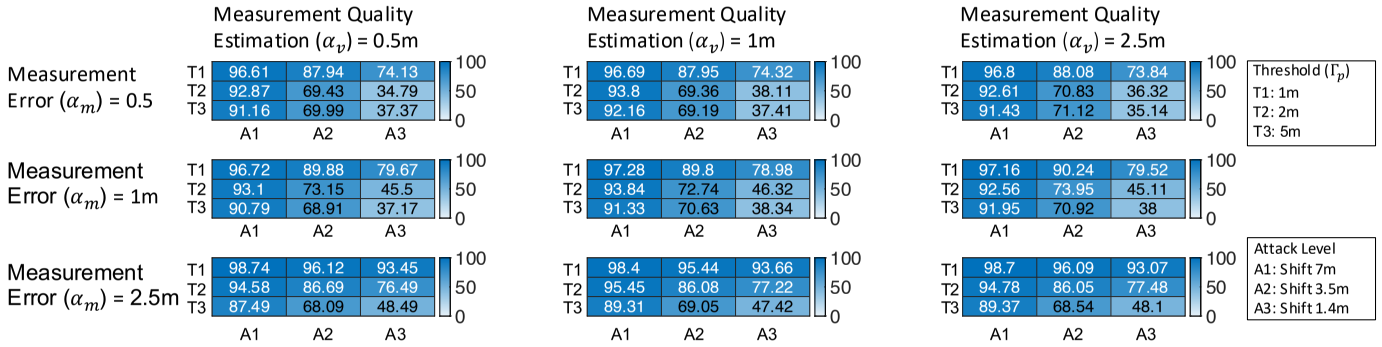


Figure 4.31: KFV/AoA's TPRs of detecting location data manipulation while missing speed data input, where the TPR values are presented in percentage.

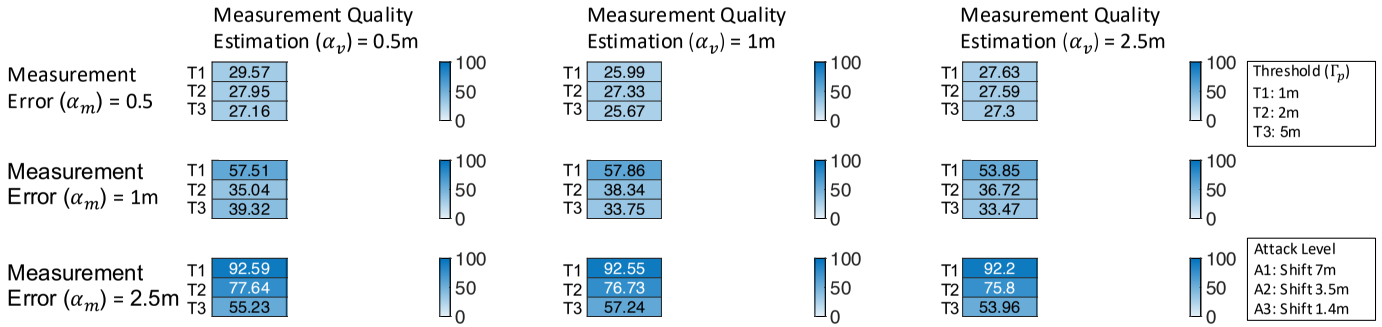


Figure 4.32: KFV/AoA's FPRs of detecting location data manipulation while missing speed data input, where the FPR values are presented in %.

Figure 4.33: KFV/AoA's detection latency of detecting location data manipulation while missing speed data input, where the DLs are presented in seconds.

4.7 Discussions

4.7.1 Limitations of EDRoad

EDRoad is not suitable for pure anomaly detection for which we do not need to identify anomalous data and restore it as mentioned in Chapter 4.5. If individual data verification and restoration are not required for the target application, we suggest use of RUSBoost [119], instead of **EDRoad**, for better FPR performance while treating all data as a single group.

Another natural limitation of **EDRoad** is that engineers must input a set of SDs/classifiers that provide sufficient correlation/causality to support its individual data verification. The sufficient condition for deterministic verification of individual data is that SDs can be transformed into an equation system with 0 degree of freedom. That is, even if the initial SDs provided by the engineer is not sufficient for **EDRoad** to generate a VT template, s/he should be able to find additional SDs derived from the initial SDs to fill in the gap of VT template generation after receiving the hint provided by **EDRoad** (Chapter 4.4.2.2). This step can also be done automatically by using an equation solver, such as Matlab or Mathematica.

In case the system cannot provide sufficient trusted inputs for individual data verification, **EDRoad** can choose to group a set of data together for constructing the VT template. That is, a non-root node will now have multiple data-types marked in it. However, these data-types will always have the same detection results.

4.7.2 Training Data Preparation

We now introduce the process that developers should follow during real-world deployment.

First, the developers should collect the (untampered with) messages from the deployment site. These (training) messages can just be the messages transmitted from developers'

vehicles.

Second, developers should estimate the error distribution based on the collected messages and their corresponding measurements made by the RSU to determine the common error levels that will be used in the training process. In our evaluation, the first two steps are replaced by using Safety Pilot Dataset and injection of measurement errors, respectively.

Third, **EDRoad** will randomly manipulate the training messages to deviate from its actual value by the amount *equal to or larger than* the detection thresholds, which are determined by the application/service requirements. Note that developers may also include additional training cases with different manipulation levels. During our evaluation of location verification, the manipulation levels we use are up to 5m for both north↔south and east↔west direction.

Finally, developers can now use the manipulated data for training.

4.8 Conclusions

We have developed **EDRoad** as an ensemble learning framework and also as a data verification system. Engineers can simply input system descriptions to establish *individual* data verification and recovery with the capability of setting desired detection targets. Specifically, **EDRoad** automatically transforms its inputs to classifiers and further generates (a set of) verification procedures for online execution. Our evaluation has shown **EDRoad** to outperform existing ensemble learning systems in detecting multi-data anomalies. An in-depth case study has shown **EDRoad** to achieve excellent detection ($>$95% TPR and $<$4% FPR) and data restoration performance (with a $<$7% average deviation from the ground truth).

CHAPTER V

CADCA: Detection and Resolution of Control Decision Anomalies

5.1 Introduction

As advanced driver assistance system (ADAS) and self-driving capabilities are gradually integrated into passenger and commercial vehicles, we have officially entered the era of semi-autonomous vehicles (SAVs), which will be the most commonly seen SA systems in the near future. In addition to enhancing passengers' comfort and reducing fuel consumption, the main reason for adopting autonomous functions in SA systmes/vehicles is to ensure passengers' safety.[1] However, autonomous controls in SA systems still have a long way to go as there can never be a perfect human-in-the-loop system that is free of design flaws which can be exploited by attackers (*e.g.*, Jeep Cherokee hack [87]). Other than attacks that directly send malicious control commands to SA systems, data manipulation or sensor spoofing can also endanger their safety. In particular, there are three factors that may jeopardize SA systems' safety:

F1. Sensor failures and falsification attacks that generate incorrect inputs to the autonomous system (*i.e.*, no data can be entirely trusted!);

F2. Compromised/imperfect controllers or algorithms that generate dangerous controls

[1] Drivers are known to be the reason for up to 95% of car accidents [94].

even with correct input; and

F3. Malicious/unsafe control inputs from the human operator.

Since neither autonomous nor manual control is perfect, a conflict or disagreement between them may arise. To resolve such a conflict, state-of-the-art safety features are usually implemented with *static* assignment of priority. For example, an automatic emergency braking can override the driver's control if an object is detected in front of the vehicle [51]. However, this static priority assignment can lead to catastrophic accidents when accompanied with incorrect sensor readings.

The crash of Boeing 737 MAX [48] is the most iconic example from which engineers must learn for the design of SA systems. In 737 MAX's original design, its Maneuvering Characteristics Augmentation System (MCAS) was given priority over pilots' control, causing pilots to compete with MCAS for the control of the plane's pitch angle when the Angle-of-Attack (AOA) sensor malfunctions, eventually leading to two fatal crashes since the pilots could not disable MCAS completely. Boeing's updated MCAS [14] addresses the above problem by using two (instead of only one) AOA sensors for data integrity verification and modifying MCAS to activate only once (*i.e.*, MCAS alone will never override pilots' control). However, this countermeasure returns the control priority back to the pilot, reversing the design to potentially allow malicious/erroneous manual control, *e.g.*, the pilot may intentionally or accidentally crash the airplane, like the Germanwings 9525 incident [108].

Since the static assignment of priority to either manual or autonomous control input has shown to fail in safety-critical situations, engineers need to answer the following critical question: *How can an SA system prevent the execution of malicious/unsafe controls (F2 & F3) under attacks and failures (F1) that feed the system incorrect inputs?*

We propose **CADCA** (Context-Aware Detection and resolution of Control Anomalies) as an effective answer to this question. As shown in Fig. 5.1 with the example of an SAV, **CADCA** is designed for use in SA systems equipped with both autonomous and manual

control capabilities. **CADCA** can be considered as a standalone decision-maker (*i.e.*, independent of the autonomous control system) that can be deployed in the ego SA system and aims at identifying the source(s) of anomalies in order to prevent use of their inputs/controls for its operation.

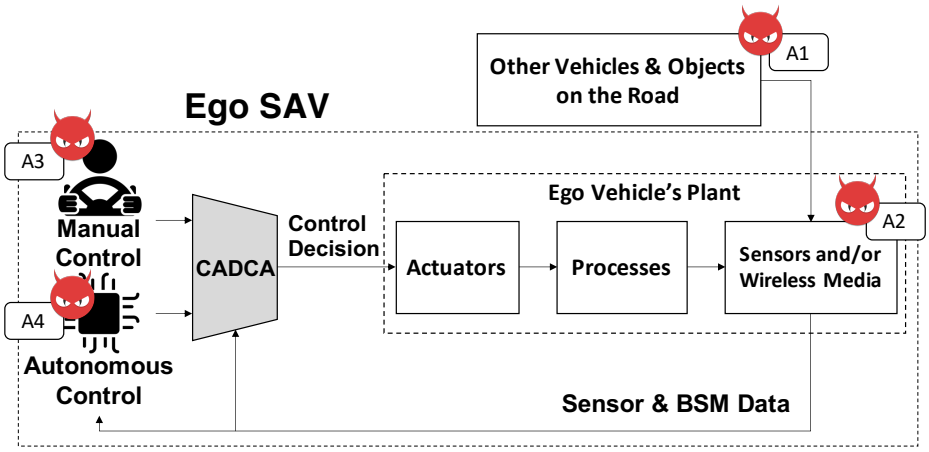


Figure 5.1: **CADCA**'s example application application.

There are two characteristics that differentiate SA systems from typical, stand-alone cyber-physical systems (CPSes):

C1. *Behavior consistency does not imply safety*: Whether a control decision is safe or not depends on its operation context, and hence checking whether i) an SA system acts consistently with the control input (*e.g.*, prior anomaly detection schemes [25, 28, 104]) or ii) the autonomous control always yields the same control decision with the same sensor input is not enough to ensure the SA system's safety. **CADCA** needs to estimate the operation "context" based on potentially anomalous data to determine if a control decision is safe to execute.

C2. *Limited observability*: The autonomous control in an SA system makes decisions based on not only its own measurements but also the information/data received from other SA systems, say, via Basic Safety Messages (BSMs), while it is the vision of the governments [134–136], car manufacturers [40, 49], and chip makers [103] that there will be vehicle-to-everything (V2X) communications, to support the Intelligent

Transportation Systems (ITSes) for future autonomous vehicles. Data received from other SA systems do not have the same level of detail in information as those measured by the ego SA system itself. Also, the ego SA system usually does not know other SA systems' initial states and control decisions that have been used/assumed by most prior work on estimation-based anomaly detection.

While prior studies of robotic vehicles or industrial control systems usually assumes observability,[2] this setting enables them to perform deterministic state estimation under anomalies/attacks while treating the process as solving a pure mathematical/optimization problem based on an equation system depicting transitions and causalities between system states and their outputs without considering the feasibility/difficulty of attacks. However, such an approach cannot work on SA systems since there can be multiple attack/anomaly scenarios all lead to the same observations due to the lack of sufficient input and observations (Chapter 5.2).

Therefore, **CADCA** employs a new design to first use the difficulties of attacks as a context to eliminate uncertainties in the equation system by identifying a (set of) probable anomaly scenario(s) and then restore the identified anomalous data in their corresponding scenarios. That way, **CADCA** inherently takes attacks' difficulties into account and avoids i) blindly consuming computing resources to find all possible (including infeasible) solutions that lead to the same observation, and ii) getting trapped with a single suboptimal/local solution that does not match the ground-truth condition.

Specifically, **CADCA** will (i) determine if there is any potential anomaly embedded in the received/perceived data, (ii) generate the most likely operation contexts that the SA system may experience, (iii) perform risk assessment for most likely scenarios, and (iv) aggregate the results from risk assessments to generate the final control decision. Using an SAV as a concrete case study, **CADCA** cross-validates the received data to check if any data can be incorrect (meeting F1) upon acquiring the ego SAV's sensor readings and

[2] Whether the internal state of the target system can be deterministically identified by the observed output.

the status reports embedded in the BSMs received from other entities in the vicinity. If any inconsistency among the received data is detected, **CADCA** will restore the anomalous data by constructing one or more local views that describe the most probable driving contexts (*e.g.*, the relative location, speed and heading of other vehicles). This way, **CADCA** can capture the conditions that the ego SAV is most likely to encounter without restricting itself to only one specific scenario. Finally, **CADCA** performs risk assessment based on *all* ⟨local view, control input⟩ combinations and aggregate their results for selecting a final control decision (meeting F2 & F3) based on a safety-first principle while ensuring maximum flexibility of drivers' control.

This chapter makes the following contributions:

- A novel control decision-maker, **CADCA**, that accounts for the difficulty of attack as a context for determining probable operation scenarios, including:
  - An efficient anomaly detection mechanism under C1 and C2 (Chapter 5.4.2);
  - A mechanism for restoring local views to help the ego SA system comprehend its operation context (Chapter 5.4.3);
  - An efficient risk assessment for a given ⟨local view, control input⟩ setting and result aggregation mechanisms under uncertain situations (Chapters 5.4.4); and
- Extensive evaluation of **CADCA** (with >15,700 test-cases), demonstrating its ability to achieve 98% success rate in avoiding use of malicious control inputs and preventing vehicle collisions in the most commonly seen driving scenarios under component failures and/or attacks (Chapter 5.5).

Similar to previous chapters, we will use SAVs to illustrate **CADCA**'s design.

5.2 Related Work

5.2.1 Risk Assessment and Collision Avoidance

Prior work on robotics and vehicle systems has explored ways of assessing risks to prevent the system from entering unsafe states [75]. Fraichard and Asama [41] proposed the concept of inevitable collision states for the analysis of navigation and motion planning. Brännström *et al.* [16] proposed a model-based algorithm to estimate how the vehicle can take actions to avoid collision with an object. Lawitzky *et al.* [73] developed a framework for predicting the motion of vehicles to assist collision avoidance. [67] proposed how to compute the trigger time for an automatic braking system to avoid at least three types of collision scenarios. Berthelot *et al.* [11] developed an algorithm for efficient computation of time-to-collision under the assumption that the received data have a multivariate normal distribution. As one of the latest risk-assessment schemes, [9] proposed the utilization of sensor fusion and inter-vehicle communications for predicting a vehicle's trajectory.

While prior approaches can efficiently perform risk assessment under the condition they are specifically designed for, they assume their data to be *always* trustworthy (*i.e.*, having bounded or known error distributions). In reality, however, this assumption does not hold and there can be sensor failures or even malicious attacks (other than measurement noises) that cause data to be erroneous. A system must, therefore, account for the situation where any of input data can be anomalous/incorrect.

5.2.2 State Estimation and Anomaly Detection

A typical approach to anomaly detection in a cyber-physical system (CPS) is to formulate the target system with the following equation system [46, 122, 150]:

$$\begin{cases} x[k+1] = A(x[k]) + B(u[k]) + \xi_p[k] + \mu[k] \\ z[k] = C(x[k]) + \xi_m[k] + \psi[k], \end{cases} \tag{5.1}$$

where $x[k]$ is the ground-truth system state with time index k, z is the observed output, u is the control input/setpoint and μ and ψ are the process and measurement noises, respectively. A, B, and C are functions/transformations used to capture the correlation/causality between system states, control inputs, and measurement outputs and ξ_p and ξ_m are the vectors depicting the effect resulting from an attack. Given the constraints of bounded process and measurement noises (*i.e.*, $\mu \leq \bar{\mu}$ and $\psi \leq \bar{\psi}$ are always true for some $\bar{\mu}$ and $\bar{\psi}$), the attack (*i.e.*, ξ's) can then be identified by solving Eq. (5.1). Note x, z, u, μ, ψ, and ξ's can all be in vector forms while A, B, and C have appropriate output dimensions. Utilizing Eq. (5.1) to have a unique, deterministic solution requires u to be known to the detection system and the CPS to have certain observability properties [122]. However, this requirement cannot be met in **CADCA**'s use case (C2) that involves multiple entities. That is, the ego SA system does not have u in the first equation of Eq. (5.1) for all non-ego SA systems in practice. Also, prior work (e.g., [122]) usually transforms Eq. (5.1) further to an optimization problem and solves it numerically assuming the existence of only one solution. This will make the process of finding the solution stop/stuck at the first (and maybe incorrect) solution it identifies without searching for other feasible ones.

There are also approaches tailored for vehicles with special sensors under multi-entity scenarios [15, 105, 116, 141]. They usually conduct a plausibility/consistency check to their data of interest by cross-validating the received data with wireless/distance measurements and motion prediction. Specifically, [63] proposed a data verification scheme based on Kalman Filter (KF) to predict vehicle locations which reports the detection of an anomaly if the prediction does not match the received data. Bißmeyer *et al.* [13] utilizes the particle-filter to cross-validate vehicles' location with radar readings to track and verify the motion of neighboring vehicles. Stübing *et al.* [127] proposed a two-stage data verification scheme for verifying position and velocity based on KF and motion prediction with probabilistic maneuver recognition.

While most, if not all, of prior anomaly detection in vehicle domain focus only on de-

tecting data anomalies, they usually consider the data of interest as a whole and neither identify which data are anomalous nor restore them, thus missing one of the most crucial functions required by **CADCA**. Also, when there is inconsistent information, multiple scenarios may all lead to the same observation by the ego vehicle as mentioned in Chapter 5.1. This is also the very reason why a direct combination of prior anomaly detection (even if it has the capability of restoring anomalous data) and risk assessment cannot work in practice. How to perform risk assessment with multiple control inputs while assuming no data can be entirely trusted (F1), despite its importance, has not yet been fully addressed.

5.3 Deployment and Attack Models

5.3.1 Deployment Model

The ego SA system is assumed to perceive/receive three types of data: i) the measurements of its own state, ii) the state report from other SA systems, and iii) its own measurements/observations of other surrounding SA systems. Specifically, we use the following equation system to capture all the observations/data the ego entity perceived/received:

$$\begin{cases} z_{(e)} = C_{(e)}(x_{(e)}) + \xi_{m(e)} + \psi_{(e)} \\ z_{(j)} = C_{(j)}(x_{(j)}) + \xi_{m(j)} + \psi_{(j)} \\ z_{(e,j)} = C_{(e,j)}(x_{(e)}, x_{(j)}) + \xi_{m(e,j)} + \psi_{(e,j)}, \end{cases} \tag{5.2}$$

where x, z, ψ, C, and ξ_m's are the same as in Eq. (5.1) while e and j represents the ego entity and non-ego entity with index j, respectively, and the subscripts (e), (j), and (e,j) correspond to the three data types mentioned earlier, respectively. The measurement noise ψ is assumed to have a certain bound $\bar{\psi}$ under normal system condition.

The default system states (x's) and measurements (z's) considered in **CADCA**'s case study of SAVs are vehicle location (p), speed (v), acceleration (a), heading (h), and yaw

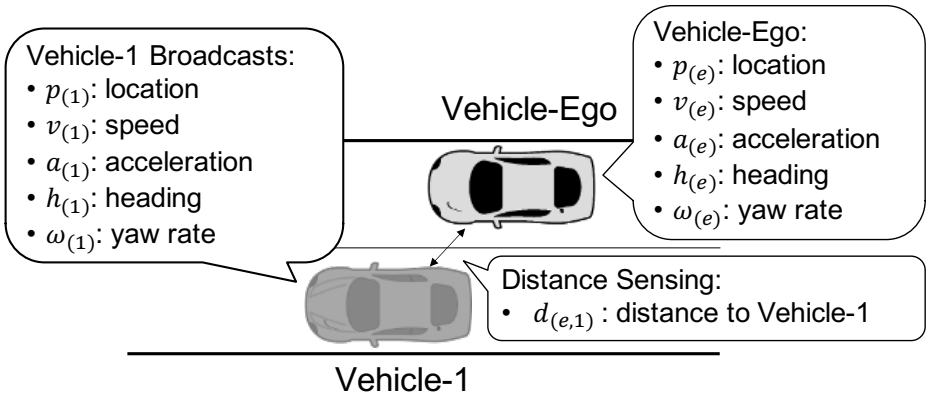


Figure 5.2: **CADCA**'s basic deployment setting.

rate (ω). The ego SAV is also capable of making measurements with its own sensors, such as RADAR, LIDAR, or cameras, to obtain the distances between itself and other SAVs (Fig. 5.2). Because x's, ξ's, and ψ's are all unknown to the ego SAV, Eq. (5.2) does *not* necessarily have a unique solution under attacks. That is, directly solving Eq. (5.2) to deterministically determine the true state (*i.e.*, x) of each SAV is not possible since there can be multiple solutions that all lead to the same observations as in Eq. (5.2).

To facilitate deployability, the information shared between vehicles is limited to the vehicle status information already embedded in standard Basic Safety Messages (BSMs) [114], including the sender vehicle's location (p), speed (v), acceleration (a), yaw rate (ω), and heading (h). That is, **CADCA** is designed to run directly on a single vehicle (*i.e.*, the *ego vehicle*) without requiring cooperation (*i.e.*, additional message exchange for obtaining computation results) from other vehicles. To avoid confusion with a single data sample, we use the term "data type" (DT) to denote a specific type of data from a specific vehicle (*e.g.*, Vehicle-1's location and Vehicle-Ego's speed).

5.3.2 Attack Model

We use SAVs as a concrete illustration of **CADCA**'s threat model. Other than malicious/erroneous human control and faulty autonomous control algorithms (A3 and A4 in Fig. 5.1), we assume the adversary has the goal of causing collision to the ego SAV *re-*

motely by i) transmitting BSMs with incorrect data from non-ego vehicles (A1) and/or ii) remotely spoofing the sensors for object recognition and distance/dynamics measurements of the ego vehicle (A2), including p, v, a, ω, and h. Specifically, **CADCA** assumes three levels of threat an adversary can pose, depending on the adversary's capabilities as shown in Fig. 5.3. These threat levels are determined by the attack attributes that are directly linked to the level of difficulty for the adversary to launch attacks.

The lowest threat level includes S1 and S2, where the (naïve) adversary can only manipulate one DT (*e.g.*, spoofing only the location/GPS of a vehicle) or manipulate multiple DTs (*e.g.*, both the location and speed of a vehicle) but s/he does not manipulate the DTs in a way that matches the normal data correlation (*e.g.*, the manipulated location and speed do not match each other). An attack is said to match normal data correlation if the data under the attack show the same correlation/causality in terms of the kinematics of rigid bodies and control-to-dynamics responses as those before the attack (within a predefined error tolerance). In the medium threat level (S3–S4), the adversary can launch an attack or spoof the DTs from a single entity and the anomalous data from that entity may be consecutively manipulated to *partially* match the normal data correlation. For example, the adversary can manipulate the location and speed in a BSM simultaneously so that the vehicle's moving distance matches the manipulated speed. In the highest threat level (S5–S6), the adversary can further expand his/her attack target to the DTs of multiple vehicles. **CADCA** is designed mainly to defend against the low and medium levels due to its assumption of no trusted data.

Specifically, **CADCA** is *not* designed to operate when *majority*[3] *or all* of the data from multiple anomalous entities are manipulated to perfectly match their normal correlation/-causality (see Chapter 5.5.8 for more analysis). Note that the above is a natural limitation of all approaches in the absence of a trusted data. However, this limitation will not diminish **CADCA**'s value for the following reason.

[3]Two-thirds of data in all anomalous entities in the case study.

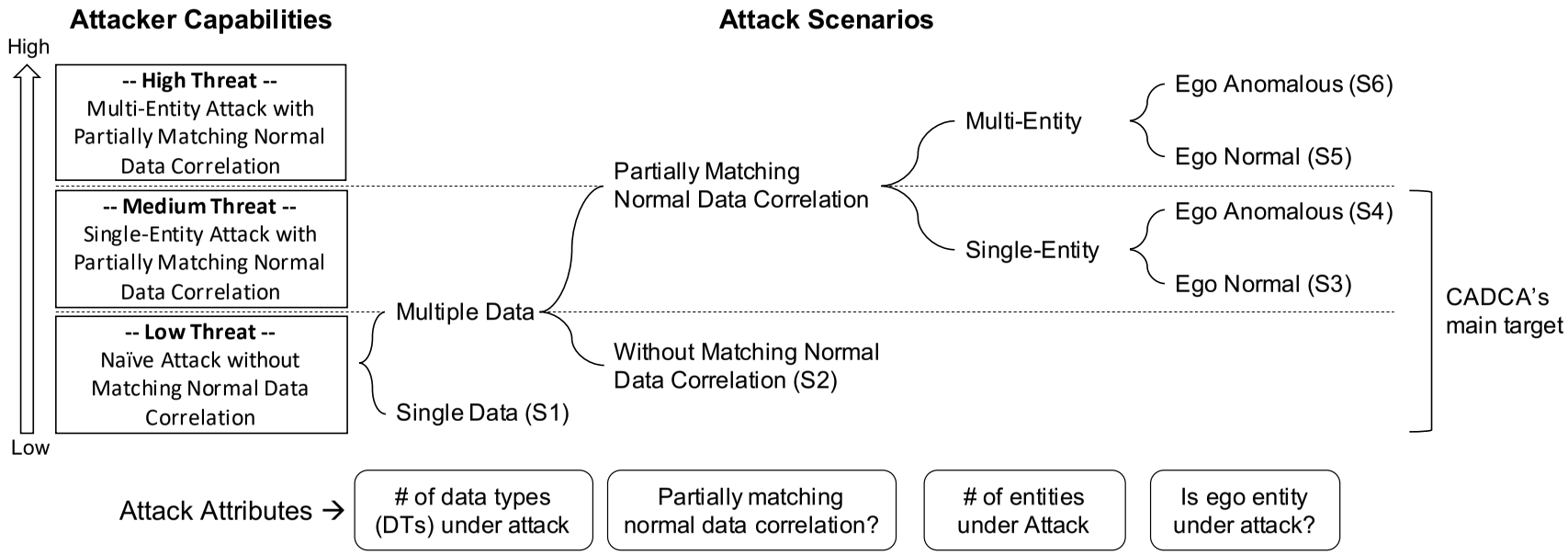


Figure 5.3: Attacker capabilities and attack scenarios considered in **CADCA**.

Unless the adversary has the complete control over non-ego vehicles' BSM transmissions, simultaneously spoofing all of their DTs (other than the location/GPS) to generate specific values is proven very difficult, if not impossible, without the direct/physical access to all the sensors. For example, spoofing a MEMS-based accelerometer requires direct feedback from the sensor for phase tuning to generate a specific waveform/value [133]. That is, even though an adversary may have the capability described in the high threat level, these types of attack will not be as scalable as the attacks in low/medium threat level since the adversary needs to physically compromise multiple vehicles.

5.4 System Design

5.4.1 Operation Overview

The lack of trusted, complete u to solve Eq. (5.2) deterministically requires **CADCA** to resolve the following questions:

Q1. How to detect a sensor failure or a data attack?;

Q2. How to identify potential operation contexts with manipulated/erroneous input data?; and

Q3. How to detect a malicious/unsafe control input under uncertain operation contexts?

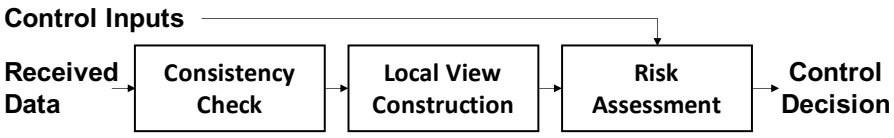


Figure 5.4: An overview of **CADCA**'s operation.

CADCA follows the design concept shown in Fig. 5.4, where each function block answers one of **Q1**–**Q3**. First, it cross-validates the data acquired by the ego entity itself and those received from other entities to detect if there is any inconsistency indicating the existence of a data anomaly or an attack. Second, upon detection of an anomaly, **CADCA** will construct local views to establish models of its surrounding environment, each which

describes a probable scenario when different combinations of data are anomalous but can lead to the same observed inconsistency. The above two steps are equivalent to first identifying the ξ's in Eq. (5.2) to eliminate the uncertainties in the equation system and then restoring the identified anomalous data based on the remaining correct ones under certain attack scenarios. Finally, **CADCA** conducts risk assessments on all probable $\langle$control input, local view$\rangle$ combinations, and then aggregates the assessment results to select a final control decision.

5.4.2 Consistency Check

CADCA's first function block is the consistency check of the input data, determining if there are any inconsistencies among the received/perceived data by cross-validating the correlation between them. While utilizing physical invariants for detecting inconsistencies is not new, **CADCA**'s novelty lies in the consideration of input data in *small DT groups*, instead of a single large group, and transformation of the result of each group's consistency-check into a corresponding equation in an equation system; solutions of the equation system will directly indicate the potentially anomalous data. We call such DT groups *Detection Sets* (DSs).

Table 5.1 shows an example of DSs when there are only two entities (*i.e.*, the ego vehicle and Vehicle-1). Specifically, $DS_{E,1}$–$DS_{E,5}$ ($DS_{V1,1}$–$DS_{V1,5}$) capture the correlation between the dynamics measurements of the ego vehicle (Vehicle-1), and $DS_{V1,6}$ describes the correlation between the distance measurement $d_{(e,1)}$ and the locations of the two vehicles. If there is another vehicle, say Vehicle-2, the ego vehicle can have 6 more observations ($DS_{V2,1}$–$DS_{V2,6}$) just like $DS_{V1,1}$–$DS_{V1,6}$. Note the DSs shown here are the default formulations used in **CADCA** that are directly derived from the kinematics of rigid bodies [57]. Since **CADCA** is designed with expandability in mind, engineers may include additional DSs in **CADCA** and the new DSs will be automatically incorporated in Local View Construction.

DS	Ego Dynamics $z_{(e)}$					Vehicle-1 Dynamics $z_{(1)}$					Dist.
	p	v	a	ω	h	p	v	a	ω	h	$d_{(e,1)}$
E,1	✓	✓	✓	✓	✓						
E,2	✓	✓									
E,3		✓	✓								
E,4				✓	✓						
E,5	✓				✓						
V1,1						✓	✓	✓	✓	✓	
V1,2						✓	✓				
V1,3							✓	✓			
V1,4									✓	✓	
V1,5						✓				✓	
V1,6	✓					✓					✓

Table 5.1: This table shows examples of detection sets of the ego vehicle and Vehicle-1, where $d_{(e,1)}$ is the distance between the two vehicles measured by the ego vehicle.

A detection set, DS_l, will have the following general form:

$$\hat{z}'[k'] = \mathcal{F}(z[k], z[k+1]), \tag{5.3}$$

where $\mathcal{F}$ is the function describing the correlation between $\hat{z}'[k']$, the prediction of a data $z' \in DS_l$ with timestamp index $k' \in \{k, k+1\}$, and other observations (*i.e.*, $z[k]$ and/or $z[k+1]$, where the latter is optional in the formulation). Note $\mathcal{F}$ can be additional observers designed by the engineers, or it can be directly derived from C's in Eq. (5.2). For example, $DS_{V1,6}$ checks whether the distance between the ego vehicle and Vehicle-1 matches the received location data and has the form of $\hat{d}_{(e,1)} = \mathcal{F}(p_{(e)}, p_{(1)}) = |p_{(1)} - p_{(e)}|_2$, where "$|\cdot|_2$" denotes the Euclidean distance and $\mathcal{F}$ equals to $C_{(e,j)}$ when we have $z_{(e,j)} = d_{(e,j)}$ in Eq. (5.2).

Definition 5.4.1. *A DS is properly designed if $z'[k'] = \hat{z}'[k'] = \mathcal{F}(z[k], z[k+1])$ is always true when there is no measurement noise or an attack in $z'[k']$, $z[k]$, and $z[k+1]$.*

During run-time, **CADCA** will use the properly-designed $\mathcal{F}$ in Eq. (5.3) to compute $\hat{z}'[k']$ and compare this prediction with the received value $z'[k']$. Specifically, if $|\hat{z}'[k'] -$

$z'[k']|| > \Gamma_{z'}$ where $\Gamma_{z'}$ is a detection threshold, **CADCA** will report an anomaly in the detection set ($DS_l \rightarrow$ ✗); otherwise, **CADCA** will report the detection set to have passed the consistency check ($DS_l \rightarrow$ ✓). $\Gamma_{z'}$'s should be set to values no larger than common error bounds to avoid miss detections. While small thresholds may cause false-positives in the consistency-check phase, it will not cause any significant influence on **CADCA** since the data determined to be anomalous will be restored in the subsequent phase.

CADCA will then take the results from the consistency check to establish an equation system $\mathcal{E}$ and its solution will tell which data may be anomalous. Each DS, say DS_l, that fails the consistency check will yield one equation:

$$\sum_{\forall z_i \in DS_l} I(z_i) \geq 1, \tag{5.4}$$

where $I(\cdot)$ is the indicator function describing whether a received data z_i is anomalous or normal:

$$I(z_i) = \begin{cases} 1, & \text{if } z_i \text{ is anomalous.} \\ 0, & \text{if } z_i \text{ is normal.} \end{cases} \tag{5.5}$$

This formulation describes the number of data that are potentially anomalous — if a DS fails the consistency check, there will be at least one anomalous data in that DS. Since DSs directly captures the data correlation based on the laws of physics, there are no false-positives *under properly designed DSs* (*i.e.*, there must be an anomaly regardless whether it is caused by an attack, a fault, or measurement noise). Note that $DS_l \rightarrow$ ✓ does not necessarily mean no anomalous data is in DS_l (*i.e.*, it may be a false-negative detection during a consistency check).

Property 5.4.2. *For every solution to the equation system ($\mathcal{E}$) constructed by Eq. (5.4) under properly designed DSs, there exists at least one corresponding solution to Eq. (5.2).*

Proof. Assume no valid solution to Eq. (5.2). $\Rightarrow$ There exists some anomalous DS_l (*i.e.*,

$DS_l \rightarrow$ ✗) s.t. $z_i[k] = C_i(x[k]), \forall z_i \in DS_l$. However, the definition of properly designed DSs tells us that DS_l will pass the consistency check since all its data do not have measurement noise or attack. This contradicts $DS_l \rightarrow$ ✗. Therefore, the assumption must be false. □

Next, **CADCA** will solve the equation system to construct local views for the subsequent risk assessment.

5.4.3 Construction of Local Views

5.4.3.1 Concept

A local view, as the name suggests, is what the ego entity thinks/estimates its current operation context to be (*e.g.*, the status of the ego SAV itself and the surrounding SAVs). See Table 5.2 for an example local view in the form of a state table. In an ideal scenario without component failures or attacks, the ego entity should be able to construct a perfect local view that matches the ground truth (with the bounded deviation caused by measurement noises). However, if the ego entity receives/perceives anomalous data, there may exist some inconsistent information in the data received or locally perceived that may prevent the ego entity's construction of a correct local view. The basic idea of Local View Construction is to i) identify potentially anomalous data and ii) correct the anomalous data, if any, based on the normal data.

Vehicle-ID	p	v	a	ω	h
Ego	(0,30)	20	0.1	0	0° (North)
2	(30, 30)	19	0	0	180° (South)
...			...		

Table 5.2: A local view example.

To identify the anomalous data that need to be restored, **CADCA** will solve the equation system ($\mathcal{E}$) to identify the data z_i with $I(z_i) = 1$. Since Eq. (5.2) may not have a unique solution, there can be multiple solutions that satisfy $\mathcal{E}$ even in a perfect detection scenario (*i.e.*, with neither false-positives nor false-negatives during the consistency check phase).

Therefore, **CADCA** constructs multiple local views, each of which corresponds to one valid solution to $\mathcal{E}$ that represents a specific attack scenario. Note that solving $\mathcal{E}$ is equivalent to the NP-Complete fitting set problem while assuming a perfect consistency-check phase and considering all probable false-negative/positive situations will be equivalent to solving multiple NP-Complete problems. Also, since not all the solutions have the same level of feasibility given **CADCA**'s application context, we have designed six (heuristic) algorithms to solve $\mathcal{E}$ for constructing local views (instead of blindly finding all probable solutions for $\mathcal{E}$).

Next, we provide an overview of these six algorithms. The Solution-Space algorithm (Chapter 5.4.3.2) is designed to capture the search space of anomalous data and provide the other 5 algorithms with a list of potential anomalous data as a starting point for identifying the actual anomalous data. The Greedy (Chapter 5.4.3.3) and Anomalous-Individual algorithms (Chapter 5.4.3.4) are designed to capture solutions to $\mathcal{E}$ that have the minimum number of anomalous DTs and entities, respectively. The Trust-Ego (Chapter 5.4.3.5), Anomalous-Ego-Only (Chapter 5.4.3.6), and Anomalous-Ego-and-Others (Chapter 5.4.3.7) algorithms are designed to capture the special scenarios related to the normality of the ego entity (*i.e.*, S3–S6). While the six algorithms do not have any obvious, one-to-one mapping associated with the six attack scenarios, they do collectively cover S1–S6 defined in Fig. 5.3, which we will introduce later.

5.4.3.2 Solution-Space Algorithm

The Solution-Space algorithm (Algorithm 3) is designed to capture the search space of anomalous data and its basic operation is summarized as follows. First, it assumes no false-negatives (*i.e.*, miss detections) in the consistency-check phase and considers a data to be normal as long as it is included in a DS that passed the consistency check (Line 8–10 in Algorithm 3). Second, those data in the DSs that fail the consistency check (denoted by *anomalous DSs*) but not ruled out by the first step will be considered potentially anomalous

Algorithm 3: Solution-Space Algorithm

```
1  Function: Solution_Space (ℰ);
   Input  : The observer equation system (ℰ = ⟨Z,Y⟩), where Z = (DS_1, DS_2, ...) is
            the definition of each DS in the equation system and Y = (y_1, y_2, ...) is
            the consistency-check result.
   Output: A list of anomalous data (ℒ).
2
3  Set N ← total number of DSs;
4  Set K ← total number of DTs;
5  Set ℒ ← {}; // Set ℒ as an empty list.
6  Initialize arr1[N][K] to FALSE;
7  Initialize arr2[K] to FALSE;
8  for i As Integer = 1 UpTo N do
9      if y_i == "✓" then
10         Set arr1[i][j] ← TRUE, ∀z_j ∈ DS_i;
11 for j As Integer = 1 UpTo K do
12     Set arr2[j] ← ⋁_{i=1}^{N} arr1[i][j]; // Column-wise ''OR'' operation to
           arr1
13     if arr2[j] equals FALSE then
14         Set ℒ ← ℒ ⋃ {z_j};
15 for i As Integer = 1 UpTo N do
16     if arr2[i][j] == TRUE, ∀z_j ∈ DS_i and y_i == "✓" then
17         Set ℒ ← ℒ ⋃ DS_i;
18 return ℒ;
```

(Line 11–14). Third, the Solution-Space algorithm will perform a sanity check (Line 15–17) to see if all anomalous DSs contain at least one anomalous data. If there is an anomalous DS that does not satisfy the above condition, then the Solution-Space algorithm will conclude the existence of a false-negative in the consistency check and consider all the data in that anomalous DS to be potentially anomalous. We call this step as *normality inversion*.

Property 5.4.3. *(Correctness) $\mathscr{L}$ is a solution space of the equation system $\mathcal{E}$ (i.e., there must exist a list of anomalous data $\mathscr{L}_S \subseteq \mathscr{L}$ that is a solution to $\mathcal{E}$), where $\mathcal{E}$ and $\mathscr{L}$ are, respectively, the input and output defined in Algorithm 3.*

Proof. (Proof by Contradiction.) Assume $\mathscr{L}$ is not the solution space of $\mathcal{E}$, indicating at least one equation in $\mathcal{E}$ cannot be covered by $\mathscr{L}$. However, the sanity check (Lines 15–

Algorithm 4: Greedy Algorithm

Function: Greedy($\mathcal{E}, \mathscr{L}$);
Input : The observer equation system ($\mathcal{E}$) and $\mathscr{L}$ = Solution_Space($\mathcal{E}$).
Output: A list of anomalous data ($\mathscr{L}_G$).

Set $\mathscr{L}_G \leftarrow \{\}$;
Set $\mathscr{L}_{DS} \leftarrow \{\}$;
for all $z_j \in \mathscr{L}$ **do**
    $\mathscr{L}_{DS} \leftarrow \mathscr{L}_{DS} \bigcup \{i\}, \forall DS_i \ni z_j$;
while $\mathscr{L}_{DS} != \{\}$ **do**
    $z_t \leftarrow$ the data $z_j \in \mathscr{L}$ that covers the most number of DSs in $\mathscr{L}_{DS}$;
    $\mathscr{L} \leftarrow \mathscr{L} \setminus \{z_t\}$;
    $\mathscr{L}_{DS} \leftarrow \mathscr{L}_{DS} \setminus \{i\}, \forall DS_i \ni z_t$;
    $\mathscr{L}_{\mathscr{G}} \leftarrow \mathscr{L}_{\mathscr{G}} \bigcup \{z_t\}$;
return $\mathscr{L}_G$;

17) of Solution-Space algorithm ensures every equation (or anomalous DS) in $\mathcal{E}$ will be covered by $\mathscr{L}$. Therefore, $\mathscr{L}$ must not be the output of Solution-Space algorithm. □

5.4.3.3 Greedy Algorithm

Greedy algorithm (Algorithm 4) is designed to find a solution to $\mathcal{E}$ that requires as few anomalous data as possible (*i.e.*, $\arg\min_{I(z_i)} \sum I(z_i)$ given Eq. (5.4) as the constraint) in a greedy way. Specifically, it will identify the anomalous data starting from the list obtained from Solution-Space algorithm (*i.e.*, $\mathscr{L}$) and pick the anomalous data z_j that covers the most number of anomalous DSs. A DS_i is said to be *anomalous* if $\exists\, z_t \in (\mathscr{L} \cap DS_i)$. Greedy algorithm will then remove all the data in $\cup_{\forall DS_i \ni z_j} DS_i$ from $\mathscr{L}$. Next, it will continue to pick the data in $\mathscr{L}$ that covers the most remaining anomalous DSs and repeat the process until all the anomalous DSs are covered.

5.4.3.4 Anomalous-Individual Algorithm

This algorithm is designed to capture a solution to $\mathcal{E}$ that requires the attacker to compromise the least number of entities. Unlike Greedy algorithm that directly selects the data

Algorithm 5: Anomalous-Individual Algorithm

Function: Anomalous_Individual($\mathcal{E}, \mathscr{L}$);
Input : The observed equation system ($\mathcal{E}$) and $\mathscr{L}$ = Solution_Space($\mathcal{E}$).
Output: A list of anomalous data ($\mathscr{L}_{AI}$).

Set $\mathscr{L}_{AI} \leftarrow \{\}$;
Set $\mathscr{L}_{DS} \leftarrow \{\}$;
Set $\mathscr{L}_V \leftarrow \{V_0, V_1, ..., V_M\}$; `// A list of all entities, where` V_0 `is the ego entity and` M `is the number of entities (excluding the ego entity).`
for all $z_j \in \mathscr{L}$ **do**
    $\mathscr{L}_{DS} \leftarrow \mathscr{L}_{DS} \bigcup \{i\}, \forall DS_i \ni z_j$;
while $\mathscr{L}_{DS}$! = {} **do**
    $V_t \leftarrow$ the entity $V_j \in \mathscr{L}_V$ that covers the most DSs in $\mathscr{L}_{DS}$;
    while $\exists\, z_j \in \mathscr{L}$ and z_j belongs to V_t **do**
        $z_t \leftarrow$ the data $z_j \in \mathscr{L}$ that covers the most DSs in $\mathscr{L}_{DS}$ and z_j belongs to V_t;
        $\mathscr{L} \leftarrow \mathscr{L} \setminus \{z_t\}$;
        $\mathscr{L}_{DS} \leftarrow \mathscr{L}_{DS} \setminus \{i\}, \forall DS_i \ni z_t$;
        $\mathscr{L}_{AI} \leftarrow \mathscr{L}_{AI} \bigcup \{z_t\}$;
        if z_j does not belong to V_t, $\forall z_j \in \mathscr{L}$ **then**
            break;
    $\mathscr{L}_V \leftarrow \mathscr{L}_V \setminus \{V_t\}$;
return $\mathscr{L}_{AI}$;

covering the most number of anomalous DSs, Anomalous-Individual algorithm will first identify the entity that covers the most number of anomalous DSs and follows the same concept of Greedy algorithm to identify the anomalous data associated with that particular entity first. Next, Anomalous-Individual algorithm will target the entity that covers the most number of anomalous DSs, excluding those already covered, and repeat the process.

5.4.3.5 Trust-Ego Algorithm

This algorithm (Algorithm 6) is designed to capture the situation in which no anomalous data originated from the ego entity. The algorithm also starts to find a list of anomalous data starting from the results obtained from Solution-Space algorithm, but it will only try to construct a list of anomalous data ($\mathscr{L}_{TE}$) from non-ego entities' data by following the

Algorithm 6: Trust-Ego Algorithm

Function: Trust_Ego($\mathcal{E}, \mathscr{L}$);
Input : The observer equation system ($\mathcal{E}$) and $\mathscr{L}$ = Solution_Space($\mathcal{E}$).
Output: A list of anomalous data ($\mathscr{L}_{TE}$).

Set $\mathscr{L}_{TE} \leftarrow \{\}$;
Set $\mathscr{L}_{DS} \leftarrow \{\}$;
for all $z_j \in \mathscr{L}$ **do**
    $\mathscr{L}_{DS} \leftarrow \mathscr{L}_{DS} \bigcup \{i\}, \forall DS_i \ni z_j$;
while $\mathscr{L}_{DS} != \{\}$ **do**
    $z_t \leftarrow$ the data $z_j \in \mathscr{L}$ that covers the most DSs in $\mathscr{L}_{DS}$ and z_j does not belong to ego entity;
    $\mathscr{L} \leftarrow \mathscr{L} \setminus \{z_t\}$;
    $\mathscr{L}_{DS} \leftarrow \mathscr{L}_{DS} \setminus \{i\}, \forall DS_i \ni z_t$;
    $\mathscr{L}_{TE} \leftarrow \mathscr{L}_{TE} \bigcup \{z_t\}$;
    if $\mathscr{L}_{DS} != \{\}$ and z_j belongs to ego entity, $\forall z_j \in \mathscr{L}$ **then**
        return NULL;
return $\mathscr{L}_{TE}$;

same concept as the one in Greedy algorithm. Note that Trust-Ego algorithm may not find a valid solution to $\mathcal{E}$ and outputs "NULL" when there is indeed anomalous data originated from the ego entity.

5.4.3.6 Anomalous-Ego-Only Algorithm

This algorithm (Algorithm 7) is designed to capture the condition in which only the ego entity is anomalous. Opposite to Trust-Ego algorithm, Anomalous-Ego-Only algorithm tries to find the solution that satisfies the observed equation system assuming the anomalous data can only originate from the ego entity. Like Trust-Ego algorithm, Anomalous-Ego-Only algorithm may also return a "NULL" solution if some non-ego entity is anomalous.

5.4.3.7 Anomalous-Ego-and-Others (AEnO) Algorithm

This algorithm (Algorithm 8) is designed to capture the situation in which the ego entity and at least one non-ego entity are anomalous. AEnO follows the same initial procedure

Algorithm 7: Anomalous-Ego-Only Algorithm

Function: Anomalous_Ego_Only($\mathcal{E}$);
Input : The observer equation system ($\mathcal{E}$).
Output: A list of anomalous data ($\mathscr{L}_{AEO}$).

Set $\mathscr{L}_{AEO} \leftarrow \{\}$;
Set $\mathscr{L}_{DS} \leftarrow \{\}$;

$\mathscr{L} =$ Solution_Space($\mathcal{E}$);

for all $z_j \in \mathscr{L}$ **do**
    $\mathscr{L}_{DS} \leftarrow \mathscr{L}_{DS} \bigcup \{i\}, \forall DS_i \ni z_j$;

while $\mathscr{L}_{DS} != \{\}$ **do**
    $z_t \leftarrow$ the data $z_j \in \mathscr{L}$ that covers the most DSs in $\mathscr{L}_{DS}$ and z_j belongs to ego entity;
    if $z_t ==$ NULL **then**
        return NULL;
    $\mathscr{L} \leftarrow \mathscr{L} \setminus \{z_t\}$;
    $\mathscr{L}_{DS} \leftarrow \mathscr{L}_{DS} \setminus \{i\}, \forall DS_i \ni z_t$;
    $\mathscr{L}_{AEO} \leftarrow \mathscr{L}_{AEO} \bigcup \{z_t\}$;
    if $\mathscr{L}_{DS} != \{\}$ and z_j does not belong to ego entity, $\forall z_j \in \mathscr{L}$ **then**
        return NULL;

as Anomalous-Ego-Only that focuses on the ego entity's data first and then moves on to examine the non-ego entities' data.

5.4.3.8 Properties and Computational Complexity

As mentioned earlier, the above algorithms are designed to collectively cover S1–S6. We now discuss their properties in the order of applicable scenarios. Specifically, Properties 5.4.4 and 5.4.12 specify the detection/identification guarantee of **CADCA**'s design, and Properties 5.4.5–5.4.7 (Properties 5.4.8–5.4.11) specify the performance guarantees and rule-out condition of S1 and S2 (S3–S6). Note that the combinations of performance guarantees and ruling-out conditions show not only **CADCA**'s "inclusiveness" in identifying the anomalous data/entities but also the "tightness" of its result (*i.e.*, **CADCA** considers

Algorithm 8: Anomalous-Ego-and-Others Algorithm

Function: Anomalous_Ego($\mathcal{E}$);
Input : The observed equation system ($\mathcal{E}$).
Output: A list of anomalous data ($\mathscr{L}_{AE}$).

Set $\mathscr{L}_{AE} \leftarrow \{\}$;
Set $\mathscr{L}_{DS} \leftarrow \{\}$;

$\mathscr{L} =$ Solution_Space($\mathcal{E}$);

for all $z_j \in \mathscr{L}$ **do**
    $\mathscr{L}_{DS} \leftarrow \mathscr{L}_{DS} \bigcup \{i\}, \forall DS_i \ni z_j$;

while $\mathscr{L}_{DS} != \{\}$ **do**
    $z_t \leftarrow$ the data $z_j \in \mathscr{L}$ that covers the most DSs in $\mathscr{L}_{DS}$ and z_j belongs to ego entity;
    if $z_t ==$ NULL **then**
        return NULL;
    $\mathscr{L} \leftarrow \mathscr{L} \setminus \{z_t\}$;
    $\mathscr{L}_{DS} \leftarrow \mathscr{L}_{DS} \setminus \{i\}, \forall DS_i \ni z_t$;
    $\mathscr{L}_{AEO} \leftarrow \mathscr{L}_{AEO} \bigcup \{z_t\}$;
    if $\mathscr{L}_{DS} != \{\}$ and z_j does not belong to ego entity, $\forall z_j \in \mathscr{L}$ **then**
        break;

while $\mathscr{L}_{DS} != \{\}$ **do**
    $z_t \leftarrow$ the data $z_j \in \mathscr{L}$ that covers the most DSs in $\mathscr{L}_{DS}$;
    $\mathscr{L} \leftarrow \mathscr{L} \setminus \{z_t\}$;
    $\mathscr{L}_{DS} \leftarrow \mathscr{L}_{DS} \setminus \{i\}, \forall DS_i \ni z_t$;
    $\mathscr{L}_{AE} \leftarrow \mathscr{L}_{AE} \bigcup \{z_t\}$;

the data/entity to be anomalous if and only if there is a reason to do so instead of blindly considering the possibility of every data/entity to be anomalous.

Property 5.4.4. *(All Scenarios: Detection Guarantee) As long as not all of the data are simultaneously manipulated to match the normal data correlation,* **CADCA** *is guaranteed to detect the anomaly.*

Proof. Due to the special DS design as shown in Table 5.1, the DSs in **CADCA** *cannot* be

partitioned into two groups, say G_A and G_B, such that

$$(\bigcup_{\forall DS_j \in G_A} DS_j) \bigcap (\bigcup_{\forall DS_j \in G_B} DS_j) = \phi \text{ (a null set)}, \tag{5.6}$$

meaning that there will always be DTs overlapping between the two DS groups. This indicates that as long as not all the data are manipulated simultaneously, there will be at least one DS containing both correct and anomalous data since there is no way to partition the data into a normal and an anomalous groups based on the DSs. □

Property 5.4.5. *(S1: Data Identification Guarantee) Greedy algorithm is guaranteed to identify the anomalous data x_a under a single-DT anomaly if no two DTs are covered by the exact same detection sets.*

Proof. (Part-I: The first identified data must be x_a.) Assume Greedy algorithm identifies another normal data x_n ($\neq x_a$) first. Since the algorithm chooses x_n first (*i.e.*, x_n must cover the same number as, or more anomalous DSs than, x_a) and no two data are covered by the exact same detection sets, there must exist some anomalous DS_k, such that $x_n \in DS_k$ but $x_a \notin DS_k$. However, this contradicts the fact that x_a is the only anomalous data since any anomalous DS must contain at least one anomalous data.

(Part-II: Greedy algorithm must terminate after identifying x_a.) If the Greedy algorithm does not terminate after selecting x_a, then there are other DSs that fail the consistency check but do not contain x_a, contradicting the fact that x_a is the only anomalous data. Thus, the assumption must be false. □

Property 5.4.6. *(S1 & S2: Entity Identification Guarantee) Solution-Space algorithm can identify the anomalous entity under a naïve attack.*

Proof. Since no false-negatives can occur under a naïve attack, no anomalous data will be ruled out by the algorithm. Therefore, the anomalous entity must be identified. □

Property 5.4.7. *(S1 & S2: Ruling-Out Condition) If the normality inversion (Line 17 of Algorithm 3) is activated in Solution-Space algorithm, then there must be a false-negative in the consistency check,* i.e.*,* **CADCA** *is dealing with an attack of medium or high level threat. (See*

Proof. Assume no false-negative occurs in the consistency check, but the Solution-Space algorithm activates the normality inversion for an anomalous DS (DS_k). Then, there must exist an x_a such that $I(x_a) = 1$ and $x_a \in DS_k$, but it is not included in the final result of the Solution-Space algorithm, and hence must be ruled out by some $DS_j \rightarrow \checkmark$. However, this contradicts the fact that DS_j must fail the consistency check under a naïve attack if $x_a \in DS_j$. Thus, there must be a false-negative detection. □

Property 5.4.8. *(S3: Ruling-Out Condition and Entity Identification Guarantee) If no solution can be found by Anomalous-Ego-Only algorithm, there must exist an anomalous data from a non-ego entity. (This can be proved by the design of the algorithm.)*

Property 5.4.9. *(S3 & S5: Entity Identification Guarantee) Trust-ego algorithm will identify any non-ego, anomalous entity/entity as long as not all the data of that entity are manipulated to match the normal data correlation.*

Proof. Assume the output ($\mathcal{L}_{TE}$) of Trust-Ego algorithm does not contain any data from an anomalous entity (E_t) under the condition in the property description. Since the Trust-Ego algorithm will not determine any data from the ego entity to be potentially anomalous and there is no data of E_t determined to be anomalous in $\mathcal{L}_{TE}$, all DSs associated with E_t must pass the consistency check, contradicting the fact that not all data of that entity are manipulated to match the normal data correlation. Thus, $\mathcal{L}_{TE}$ must contain at least one data from E_t. □

Property 5.4.10. *(S3 & S5: Ruling-Out Condition) If Trust-Ego algorithm cannot find a solution, the ego entity must be anomalous. (This can be proved by the design of the algorithm.)*

Property 5.4.11. *(S4 & S6: Entity Identification Guarantee) Anomalous-Ego-and-Others algorithm is guaranteed to identify the anomalous ego entity and any anomalous entity (E_t) if not all E_t's status data (*i.e.*, the data excluding the distance measurement) are manipulated to match the normal data correlation.*

Proof. Since not all E_t's status data are manipulated to match the normal data correlation, there must exist some DS_k associated only with E_t and $DS_k \rightarrow \text{✗}$. Therefore, AEnO algorithm must identify at least one anomalous data $x_a \in DS_k$ from E_t to cover DS_k. □

Property 5.4.12. *(All Scenarios: Entity Identification Guarantee) Anomalous-Individual algorithm is guaranteed to include the anomalous entities as long as not all the data/measurement of the anomalous source are manipulated coordinately. (This property can be proved by a similar procedure as the proof of Property 5.4.11.)*

Computational Complexity. *All algorithms have an $O(NK)$ computational complexity, where K and N are the numbers of data types per vehicle and DSs, respectively.*

Note that the above properties are the foundation of **CADCA**'s ability to determine anomalous entities in the risk-assessment phase (Chapter 5.4.4).

5.4.3.9 Restoration of Anomalous Data

The final step in Local View Construction is to restore (or estimate) data identified as anomalous and construct local views for each algorithm's output. Specifically, **CADCA** will directly utilize the data correlation captured in the detection sets for restoring anomalous data if enough correct data are observed. Otherwise, **CADCA** will use the last (set of) data determined to be correct to build the local view.

5.4.4 Risk Assessment

5.4.4.1 Basic Concept

The process of risk assessment will be turned on whenever an anomaly is detected or **CADCA** receives more than one control input. Its goal is to identify whether there exists any safety risk (*i.e.*, possible collision) associated with the control inputs, and helps the ego entity choose which of the control inputs to accept and execute. Adapting from the common practice of collision warning/avoidance systems, **CADCA**'s case study deems the existence of a safety concern if the estimated Time-To-Collision (TTC) is smaller than a threshold $T_C = \max(T_S, T_U + T_R)$, where T_S(=4.5s) is determined by the (medium) driver's reaction time while ensuring a smooth transition of control (*e.g.*, no sudden braking all the time) [17], T_U is the minimum time required for the ego entity to avoid the collision through speed control, and T_R(=2.6s) is the driver's reaction time in an urgent situation [128]. T_U is computed based on i) the relative speed $v'_{(\eta)} = v_{(\eta)} - v_{(e)}$ between the ego and the target entities η, ii) η's acceleration $a_{(\eta)}$, iii) the ego vehicle's maximum acceleration $a^+_{(e)}$ and deceleration $a^-_{(e)}$ capability, and iv) the relative location $p'_{(\eta)}$ of η with respect to ego vehicle's travelling direction:

$$T_U = \begin{cases} |v'_{(\eta)}|/(|a^-_{(e)}| + a_{(\eta)}), \text{ if } v'_{(\eta)}, p'_{(\eta)} < 0; \\ |v'_{(\eta)}|/(|a^+_{(e)}| - a_{(\eta)}), \text{ if } v'_{(\eta)}, p'_{(\eta)} > 0; \\ 0, \text{ otherwise.} \end{cases} \tag{5.7}$$

Note that Eq. (5.7) does not consider every possible combination of $\langle v'_{(\eta)}, a_{(\eta)}, p'_{(\eta)} \rangle$ because computation of T_U is required only when there is a possibility of collision.

Unlike prior studies that assume a fixed input scenario, **CADCA** is designed to operate in an uncertain situation (*i.e.*, there are multiple probable operation contexts). While **CADCA** generates a local view for each of the algorithm's result, it also performs risk assessment for every ⟨local view, control input⟩ pair. Specifically, **CADCA** performs risk

assessments for each control input by checking whether TTC $\leq T_C$ if i) the ego entity executes the control input and ii) other entities maintain their current maneuvering behavior. **CADCA** then determines which control input is safe to execute based on the following (safety-first) rules:

Rule-1: If there is no safety concern in any of the combinations, execute the manual control input.

Rule-2: If only one control input has a safety concern (in any of the probable local views), execute the other safe one.

Rule-3: If both control inputs have safety concerns, select the control input with a longer time-to-collision under the most likely local view (see below).

Specifically, **CADCA**'s local view construction is equivalent to identifying the potential states of the ego entity in the state space and **CADCA**'s risk assessment is equivalent to determining which control input can potentially lead the ego entity to the unstable/unsafe state. Next, **CADCA** will try to avoid any potential safety risk if possible; otherwise, it will try to delay the occurrence of a safety-critical event as much as possible according to the most likely condition.

5.4.4.2 Determine the Most Likely Local View

CADCA determines the local view that is most likely to capture the actual situation (*i.e.*, the "most likely" local view) based on a record of whether an entity has been determined to be potentially anomalous before. Table 5.3 shows an example of this history record. Specifically, each row of the record shows if an entity is determined to be potentially anomalous (normal) at some time t_k, and the entry will be marked with 1 (0) under the t_k column.

CADCA will consider a non-ego entity to be potentially anomalous if any algorithm result (excluding the result from Solution-Space algorithm) determines the entity to be anomalous. In contrast, the ego entity will be determined to be anomalous if Trust-Ego

entity	t_1	t_2	t_3	t_4	t_5	t_6	t_7	...	$\text{WAC}_{k,i}$
Ego	0	0	0	1	1	1	1	...	0.5
Entity-1	-	-	-	0	1	1	0	...	0.1
Entity-2	1	1	1	1	1	1	1	...	1.0
...					...				

Table 5.3: This table shows an example of entity anomaly history, where $\text{WAC}_{k,i}$ is the WAC of the entity i at time t_k.

algorithm cannot find a solution (Property-5.4.10) and it will be determined to be normal if AEnO algorithm cannot find a solution. Otherwise, its integrity will be determined by the Greedy algorithm. The rationale behind treating the ego entity differently is: i) the observed equation system is centered around the ego entity and ii) Trust-Ego, Anomalous-Ego-Only and AEnO algorithms are specifically designed to target special cases related to the ego entity. Note that the "inclusiveness" and "tightness" in the above rules for determining anomalous entities is grounded on the collective effect of the rule-out conditions and detection properties (*i.e.*, Properties 5.4.6–5.4.12) introduced in Chapter 5.4.3.8. Specifically, the five algorithms of **CADCA** can consistently achieve a $>$99% identification rate in detecting anomalous entities under low-level threats even with imperfect detections and an $>$80% inclusion rate in identifying the anomalous entities under sophisticated attacks in medium- and high-level threats (Chapter 5.5).

To identify the most likely scenario, **CADCA** summarizes the history of each entity and uses a scalar to represent the likelihood of each entity being anomalous. Since this scalar is computed based on a weighted sum of anomaly counts, we call it *Weighted Anomaly Count* (WAC):

$$\text{WAC} = \min(W_A + W_R, 1) \in [0,1], \tag{5.8}$$

where $W_A \in [0,1]$ is the weighted sum computed over all the history and W_R is the additional factor/weight used to account for the most recent anomaly detections. Specifically, $W_A \in [0,1]$ is the ratio of the number of anomalies (N_A) to the total number of records (N_T). For

example, since the ego entity in Table 5.3 has 7 records, four of which are marked to be anomalous (=1), W_A of the ego entity will be $4/7 \approx 0.57$. To avoid unstable W_A when a entity just enters the ego entity's communication range, we design **CADCA** to compute W_A normally if there are $\geq N_{min}$ records; otherwise, **CADCA** will fill in the "missing" records with 1 (*i.e.*, assuming the entity cannot be trusted initially):

$$W_A = \begin{cases} (N_A + N_{min} - N_T)/N_{min}, \text{ if } N_T < N_{min} \text{ ;} \\ N_A/N_T, \text{ if } N_T \geq N_{min} \text{ .} \end{cases} \tag{5.9}$$

W_R controls the influence of most recent N_R records:

$$W_R = (1 - \alpha^{N_{A,R}})/(1 - \alpha^{N_R}), \tag{5.10}$$

where $N_{A,R}$ is the number of anomaly reports within the latest N_R records and α is the parameter designed to adjust how fast W_R should increase if an anomaly report is received. This design ensures that W_R will i) increase while more and more "1" records are received ii) gradually (but not drastically) decrease while the entity is determined to become normal again before a certain number of "0" records are received.

The last step of identifying the most likely local view is to find a solution obtained from the algorithms that has the best match with the WACs. Specifically, we can present the WACs as a vector $\vec{W}_k = (\text{WAC}_{k,e}, \text{WAC}_{k,V1}, \ldots)$, where $\text{WAC}_{k,i}$ is the WAC of entity i at time t_k, and this vector represents a snapshot of the entities' normality perceived by the ego entity. Similarly, we can present the solutions/results from the algorithms in a vector form $\vec{\phi}_{k,\mathcal{A}} = (b_{(e)}, b_{(1)}, b_{(2)}, \ldots)$, where $b_{(j)}$ is the normality of entity j determined by algorithm $\mathcal{A}$ and $b_{(j)} = 1$ (0) indicates the entity is anomalous (normal). For example, if Greedy algorithm determines that the ego entity and Entity-2 are anomalous and Entity-1 is normal, its vector form will be $\vec{\phi}_{k,\mathcal{G}} = (1,0,1)$. While the most common way to compare the similarity of two vectors, say $\vec{\phi}_1$ and $\vec{\phi}_2$, to a target vector ($\vec{W}$) is to compare their pair-

wise inner products (*i.e.*, $\vec{W} \cdot \vec{\phi}_1$ and $\vec{W} \cdot \vec{\phi}_2$), we need to ensure that the vectors are properly normalized; otherwise, the vector with a larger magnitude will have an unfair advantage. Therefore, we can use the following formula to compute the similarity (S) between the entities' normality perceived by the ego entity and the result of each algorithm:

$$S_{k,\mathcal{A}} = (\vec{W}_k \cdot \vec{\phi}_{k,\mathcal{A}})/|\vec{\phi}_{k,\mathcal{A}}|. \tag{5.11}$$

Finally, **CADCA** selects the local view with the largest S as the most likely local view L_M and recommends the control input that has the largest TTC under L_M if both of the controls are determined to be unsafe.

5.5 Evaluation

5.5.1 Experimental Settings

We evaluate **CADCA** based on the case study of SAVs. Since **CADCA** is designed to operate under safety-critical conditions with sensor failures or malicious attacks, we use Simulink with the automated driving toolbox [84] to evaluate **CADCA**'s performance. This simulation setting allows us to explore a wide range of dangerous scenarios and traffic conditions, as car-makers commonly do for the evaluation of their algorithms before conducting final field-tests [34, 106, 109]. Specifically, the ego vehicle is equipped with a front camera for object recognition (including lane-mark detection) and radar for distance sensing. The autonomous system controls both the steering and acceleration of the ego vehicle. All vehicles periodically broadcast BSMs to inform other vehicles of their location, speed, acceleration, heading, and yaw rate. To account for noisy sensor measurements in the real world, the ego vehicle's radar is assumed to have a maximum detection range of 174m, range resolution of 2.5m, 90% detection probability and false alarm rate of 10^{-4}%. The ego vehicle's camera generates 480×640 image frames every 0.1s and its object recognition algorithm has a 90% detection rate [56]. Note that the above settings will be altered to

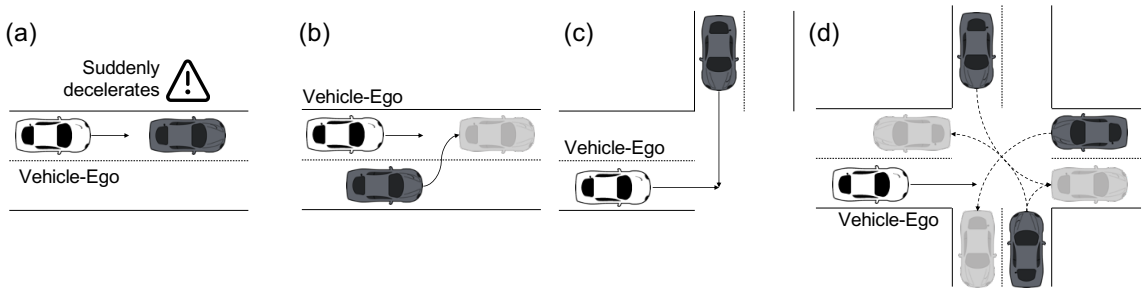


Figure 5.5: **CADCA**'s basic testing scenarios: (a) T1: front vehicle suddenly decelerates, (b) T2: driving scenarios involving (unsafe) lane changes, (c) T3: a potential side collision situation, and (d) T4: an (unsafe) encounter when Vehicle-Ego enters an intersection. Note that we omitted different potential maneuvers of the ego vehicle in the figures.

describe the conditions where the sensors are anomalous, or under the influence of a severe weather condition, yielding a much worse object recognition performance.

Next, we introduce how to generate the test cases. The relative movements of two moving SAVs can be categorized into either moving parallelly or non-parallelly. Therefore, there will be four combinations of relative movements before a collision between two entities can happen: i) Parallel Only, ii) Parallel $\rightarrow$ Non-Parallel, iii) Non-Parallel Only, and iv) Non-Parallel $\rightarrow$ Parallel. Based on this observation, we evaluate **CADCA**'s ability of preventing unsafe controls based on the commonly seen (traffic) scenarios depicted in Fig. 5.5 that collective cover all four combinations of relative movement types:

T1: (Parallel Only) As the most crucial scenario for collision avoidance, a vehicle in front of the ego vehicle suddenly brakes;

T2: (Parallel $\rightarrow$ Non-Parallel) The ego vehicle involves in an unsafe lane change;

T3: (Non-Parallel Only) A potential side-collision situation; and

T4: (Non-Parallel $\rightarrow$ Parallel, Parallel $\rightarrow$ Non-Parallel) At an intersection, the ego vehicle may experience different safety-critical encounters with vehicles from other directions.

In each scenario, there are 1–5 non-ego vehicles in the vicinity of the ego vehicle. Since we will directly introduce different controls to the ego vehicle during our experimentation, the specific motion of the ego vehicle is not explicitly listed here. We evaluate the conditions

in which both autonomous and manual controls can lead to a collision. Specifically, each test case has a unique ⟨scenario, control input magnitude, control timing, attack/anomaly⟩ combination. In each test case, at least one control input will lead to a collision. Also, the attack will start right after a test case begins and last until a collision occurs or until the test case ends. See Table 5.4 for the statistics of the test-cases in each part of the evaluation. We also implemented a baseline approach "RA-BSM", which is adapted from [9] (*i.e.*, one of the most recent risk assessments) that also utilizes both vehicle state estimation and inter-vehicle communications for its risk assessment.

Scenario	Unsafe Control	Should-Block	Should-Allow	Total
Low Threat: Single-DT Manipulation (Chapter 5.5.4)				
T1	Manual	582	822	1,404
T2	Manual	42	102	144
T3	Autonomous	42	9	51
T4	Autonomous	45	99	144
Subtotal	-	711	1,032	1,743
Medium Threat: Attacks with Correlation Matching (Chapter 5.5.5)				
T1	Manual	935	1,405	2,340
T2	Manual	70	170	240
T3	Autonomous	70	15	90
T4	Autonomous	45	75	120
Subtotal	-	1,120	1,665	2,785
High Threat: Sensor Failures with Attacks (Chapter 5.5.6)				
T1	Autonomous/Manual	1,160	5,265	6,425
T2	Autonomous/Manual	270	2,670	2,940
T3	Autonomous	595	260	855
T4	Autonomous	75	960	1,035
Subtotal	-	2,100	9,155	11,255
Total	-	3,931	11,852	15,783

Table 5.4: This table shows the number of cases tested, where each case has a 0–30s lead time to the safety-critical situation.

CADCA's performance is evaluated using two metrics:

• *Success Rate* (SR): the probability of successfully blocking the execution of a control

input that will lead to a collision when TTC $\leq T_C$. The unsafe input can be either manual or autonomous control.

• *Incorrect Blockage Rate* (IBR): the probability of unnecessarily disallowing a safe control input. Note that this metric is considered only when both manual and autonomous controls are safe, and thus the manual control should be allowed.

Note that **CADCA** does not treat the two types of controls differently if any of them is deemed to be unsafe. Therefore, **CADCA**'s performance for blocking unsafe inputs will not be influenced by which of the control types is unsafe in each test case.

5.5.2 Implementation: Detection Sets

We introduce the detailed formation of DSs used in the evaluation. We use j to indicate the DS's target vehicle. Note that j can be both the ego and non-ego vehicles for $DS_{j,1}$–$DS_{j,5}$ and it can only be a non-ego vehicle for $DS_{j,6}$.

5.5.2.1 $DS_{j,1}\{p,v,a,\omega,h\}$

$DS_{j,1}$ captures the correlation between two consecutive vehicle locations, $p_k = (p_k^{(X)}, p_k^{(Y)})$ and $p_{k+1} = (p_{k+1}^{(X)}, p_{k+1}^{(Y)})$, based on vehicle speed (v_k), acceleration (a_k), yaw rate (ω_k) and heading (h_k), where the scripts k and $k+1$ are the (timestamp) indices of the received data. Specifically, $DS_{j,1}$ has the following formulation:

$$p_{k+1}^{(X)} = p_k^{(X)} + \frac{v_k}{\omega_k}[\sin(h'_k) - \sin(h_k)] + \frac{a_k \Delta t}{\omega_k}\sin(h'_k) - \frac{a_k}{\omega_k^2}[\cos(h_k) - \cos(h'_k)], \tag{5.12}$$

$$p_{k+1}^{(Y)} = p_k^{(Y)} + \frac{v_k}{\omega_k}[\cos(h_k) - \cos(h'_k)] - \frac{a_k \Delta t}{\omega_k}\cos(h'_k) - \frac{a}{\omega_k^2}[\sin(h_k) - \sin(h'_k)], \tag{5.13}$$

$$h'_k = h_k + \omega_k \Delta t, \tag{5.14}$$

where Δt is the time interval between p_k and p_{k+1}.

5.5.2.2 $DS_{j,2}\{p,v\}$ and $DS_{j,3}\{v,a\}$

$DS_{j,2}$ ($DS_{j,3}$) captures the correlation between the vehicle speed and location (acceleration):

$$DS_{j,2} : v_k = |p_{k+1} - p_k|/\Delta t; \text{ and} \tag{5.15}$$

$$DS_{j,3} : v_{k+1} = v_k + a_k \Delta t. \tag{5.16}$$

5.5.2.3 $DS_{j,4}\{\omega,h\}$ and $DS_{j,5}\{p,h\}$

$DS_{j,4}$ ($DS_{j,5}$) captures the correlation between the vehicle heading and yaw rate (location):

$$DS_{j,4} : h_{k+1} = h_k + \omega_k \Delta t; \text{ and} \tag{5.17}$$

$$DS_{j,5} : h_k = \text{direction from } p_k \text{ to } p_{k+1}. \tag{5.18}$$

5.5.2.4 $DS_{j,6}\{p_{(e)}, p_{(j)}, d_{(e,j)}\}$

$DS_{j,6}$ captures the distance $d_{(e,j)}$ between the ego vehicle e and a non-ego vehicle j:

$$d_{(e,j)} = |p_{(e)} - p_{(j)}|_2. \tag{5.19}$$

5.5.2.5 Thresholds

As mentioned in Chapter 5.4.2, the consistency-check thresholds should set to values no smaller than their typical estimation/measurement error bound. We use the following threshold settings for **CADCA**'s evaluation (Chapter 5.5):

- Γ_p = 3m, where locations obtained from GPS has typical error of 4.9m [52];
- Γ_v = 5m/s, where the speed estimation (based on consecutive GPS readings with 1s

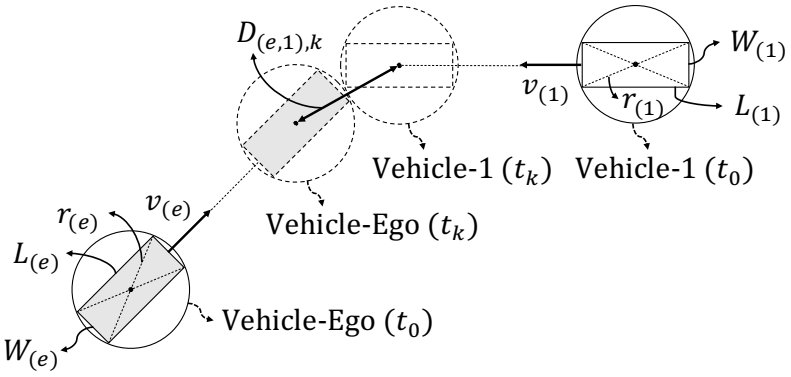


Figure 5.6: An example of computing TTC, where $D_{(e,1),k}$ is the distance between the centers of the two vehicle circles at time t_k.

update rate) can have 9.8m ($= 4.9 \times 2/1$) error;

- $\Gamma_h = \pi/16$ rad, where the heading estimation based on consecutive GPS readings can have up to $\pi/4$ rad ($= \tan^{-1}(4.9/4.9)$) error without having an orthogonal or opposite direction; and
- Γ_d = 1m (based on the distance sensing in the simulation).

5.5.3 Implementation: Estimation of Time-to-Collision

To reduce the computation work load and account for the possibility that the ego vehicle may not have the accurate information of (other) vehicles' dimensions, **CADCA** treats each vehicle as circles, instead of rectangles, in a 2-D plane and the diameters the circles are required to cover the vehicle body. Fig. 5.6 shows an example where the circle of the vehicles can just cover the vehicle body, *i.e.*, $(2r_{(e)})^2 = L_{(e)}^2 + W_{(e)}^2$ and $(2r_{(1)})^2 = L_{(1)}^2 + W_{(1)}^2$. Note that the vehicles' dimensions (*i.e.*, lengths and widths) can be obtained from BSM Part-I or estimated from the ego vehicle's LIDAR or camera.

CADCA will then predict the future trajectory of vehicles using the following equations:

$$p_{k+1}^{(X)} = p_k^{(X)} + \frac{v_k}{\omega_k}[\sin(h'_k) - \sin(h_k)] + \frac{a_k \Delta t}{\omega_k}\sin(h'_k) - \frac{a_k}{\omega_k^2}[\cos(h_k) - \cos(h'_k)], \tag{5.20}$$

$$p_{k+1}^{(Y)} = p_k^{(Y)} + \frac{v_k}{\omega_k}[\cos(h_k) - \cos(h'_k)] - \frac{a_k \Delta t}{\omega_k}\cos(h'_k) - \frac{a}{\omega_k^2}[\sin(h_k) - \sin(h'_k)], \tag{5.21}$$

$$v_{k+1} = v_k + a_k \Delta t, \tag{5.22}$$

$$h_{k+1} = h'_k = h_k + \omega_k \Delta t, \tag{5.23}$$

where subscripts k and $k+1$ are the timestamp indices of data, and $p^{(X)}$ and $p^{(Y)}$ represent the vehicle's X (east-west) coordinate and Y (north-south) coordinate, respectively. Finally, **CADCA** can determine TTC by checking the timing when the distance between the ego vehicle and another vehicle is less than the sum of their circles' radii (*e.g.*, $D_{(e,1),k} < r_{(e)} + r_{(1)}$ in Fig. 5.6).

5.5.4 Low Threat: Manipulation of a Single Data Type (DT)

We first evaluate **CADCA**'s performance when there is only one anomalous DT (the low-level threat in Fig. 5.3) with (natural) measurement noises embedded in the sensor data. This type of attack tries to trick the ego vehicle to assume a non-ego vehicle is traveling at an incorrect speed (with the value deviation Δv = 5–15 m/s) by manipulating/compromising the BSM transmission of the non-ego vehicle. We will henceforth use "VSC" to denote the vehicle that will cause a safety-critical situation and, unless specified otherwise, VSC will also be the victim of an attack (*i.e.*, VSC's data perceived by the ego vehicle can be incorrect or manipulated).

Under each of T1 – T4 scenarios, we design test-cases where the ego vehicle's au-

tonomous system may generate both safe and unsafe controls (due to imperfect control algorithm design and sensors' blind spots) and further inject manual controls that may either cause or avoid collision. The speed manipulations are designed to mislead a vehicle that performs risk assessment directly based on the received data (*i.e.*, without **CADCA**) to believe it has more time to react to the potential collision than that is actually available. Specifically, in T1, the ego vehicle is travelling at 36-108 km/h when the VSC in front of the ego vehicle suddenly stops, but the speed of VSC perceived by the ego vehicle will be larger than the ground truth by Δv. Similar to T1, the speed information embedded in VSC's BSMs will be larger than the ground truth by Δv in T2, misleading the ego vehicle to believe that the VSC will have an additional leeway to perform a lane change. To create unsafe conditions in T1 and T2, we inject manual controls that make the ego vehicle accelerate with 1 m/s^2 when it needs to decelerate to avoid a collision with VSC.

In T3, the VSC will reach the intersection at the same time as the ego vehicle, but the attack will make the ego vehicle assume VSC is moving towards the intersection at a speed deviating from its ground truth by Δv. Finally, VSC in T4 will make an unsafe right or left turn at the intersection and the attack will make the ego vehicle assume VSC will finish its turn earlier. To correct the unsafe decisions made by the autonomous control in T3 and T4, we inject manual controls to decelerate (with -2 m/s^2) the ego vehicle in order to delay its entrance to the intersection.

Fig. 5.7 shows the SRs and IBRs of **CADCA** and RA-BSM. One can observe that **CADCA** is able to achieve $\geq$92.86% SR (T2) in disallowing unsafe controls and the level of data manipulation does not have any significant impact on **CADCA**'s performance. **CADCA** is further shown to achieve $>$98% SR for T1, T3 and T4. On the other hand, RA-BSM can only achieve 55.15% SR and its performance further deteriorates when the perceived data deviates more from the ground truth, indicating it cannot properly perform risk assessment when there is a data anomaly. Furthermore, **CADCA** is able to achieve 0% IBR in all the test-cases, indicating **CADCA** will not unnecessarily disallow any of the safe

controls. That is, **CADCA** will allow drivers to manually control the vehicle if the control input is determined to be safe as if there were no additional "control-filtering" added to the ego vehicle.

5.5.5 Medium Threat: Correlation Matching

Let us consider the condition in which the attacker may manipulate VSC's (entire) BSMs while targetting VSC's location data to prevent the ego vehicle from performing accurate risk assessment. The data in VSC's BSMs after an attack will perfectly match the dynamics correlation depicted in the DSs, leaving no trace of BSM tampering. Note that the data manipulation considered here can be viewed as the final stage of Drift-with-Devil (DwD) attacks proposed in [120], which is one of the strongest attacks against the approaches utilizing state estimation. Specifically, VSC's reported location in its BSMs will be ΔX away from its ground truth and ΔX will maintain at a constant level once the test case begins. Furthermore, other data types in VSC's BSMs will match their ground truth values. Similar to Chapter 5.5.4, the attacker will try to make the ego vehicle think it has a larger leeway (*i.e.*, the distance and time) than the ground truth to reacts to the potential collision. In T1 and T2, VSC will manipulate locations in its BSMs to locations that are ΔX away from their ground truths along the travelling direction of the ego vehicle. In T3 and T4, the manipulated VSC locations will be ΔX away from their ground truths in the opposite direction of VSC's travelling direction. We use the same manual control design as in Chapter 5.5.4 to create or eliminate the unsafe conditions. In this set of evaluations, **CADCA** will face cases when it cannot deterministically identify anomalous data.

Fig. 5.8 shows the performance of **CADCA**, where the manual controls are injected when VSC's location drifts away from its actual location by ΔX = 10–25m. **CADCA** is shown to achieve >92% SR in T1 and T3 while achieving 42–64% SR in T2 and T4. The lower SRs in T2 and T4 are caused by the fact that it is harder for **CADCA** to predict and reconstruct the maneuvering behavior of vehicles since the manipulated BSMs do not

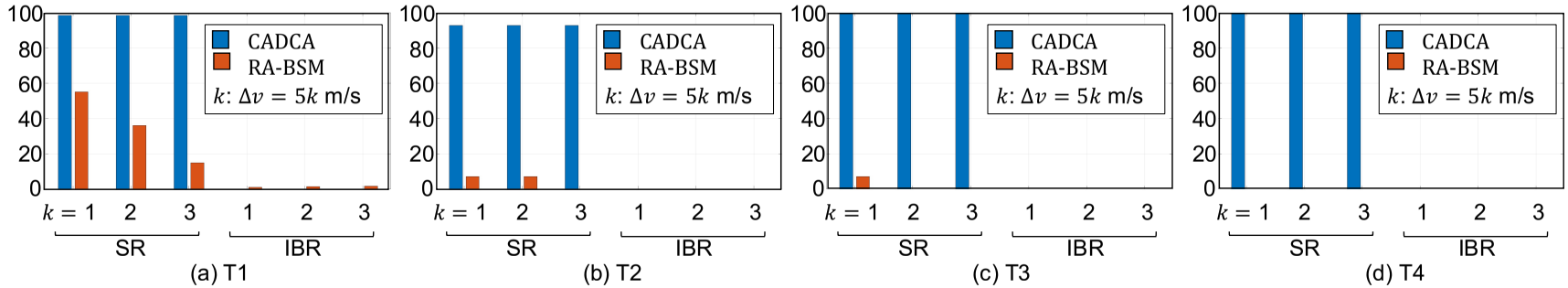


Figure 5.7: Performance (%) comparison of **CADCA** (left/blue bars) and RA-BSM (right/red bars) under single-DT anomaly. Note the missing bars indicate their corresponding values are 0%.

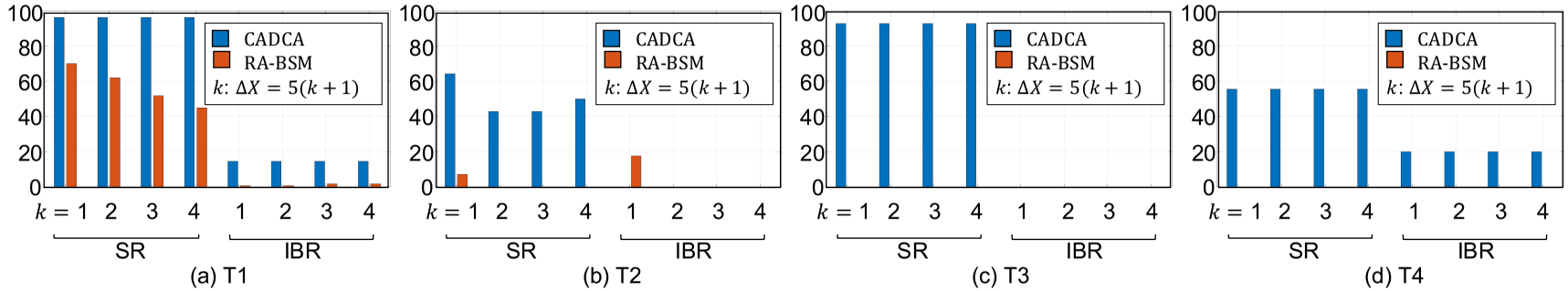


Figure 5.8: Performance (%) comparison of **CADCA** (left/blue bars) and RA-BSM (right/red bars) under multi-DT attacks targeting VSC's location, where the location deviation (ΔX) is presented in meters. Note the missing bars indicate their corresponding values are 0%.

contain any useful information that reveals the actual VSC's maneuvering behavior. That is, **CADCA** cannot accurately predict VSC's lateral movement based on the previously received data. When **CADCA**'s IBR is compared with those under single-DT manipulation, the IBRs in Fig. 5.8 are slightly larger due to **CADCA**'s safety-first design to always execute the safer control decision under an uncertain situation. Note that even though **CADCA** can only achieve moderate SRs in T2 and T4, it still has a $>$40% (absolute) SR advantage over RA-BSM.

5.5.6 High Threat: Sensor Failures and Attacks

Next, we investigate **CADCA**'s performance in the presence of both sensor/algorithm failures and attacks. Specifically, we assume the attacker may tamper with VSC's location data (ΔX = 5–25m) along with other data just like in Chapter 5.5.5 and *simultaneously* spoof the sensors/algorithms of the ego vehicle using certain adversarial machine learning techniques. To simulate this extreme condition, we purposely adjust the object detection rates of RADAR/camera to 0.9, 0.5 and 0.1 and assume there can be measurement errors (Δd = 6–10m) for the ego vehicle's distance sensing. Note that the degradation of sensing quality can also be the result of severe weather condition. **CADCA** is shown to achieve 90.43–98.33% SR and $\leq$2.44 IBR in T1 while RA-BSM can only achieve $\leq$87.83 SR and $\leq$3.83 IBR (Table 5.5). We use the same set of ground-truth vehicle behaviors within a $\langle$scenario, detection rate$\rangle$ case to capture the effects of anomalous distance sensing on **SafeSA**'s performance. Specifically, it is worth noting that **CADCA** does not experience any noticeable performance degradation even if Δd increases from 6 to 10, indicating **CADCA**'s effective resolution of control conflicts irrespective of how much error is embedded in the data once the anomaly is detected. RA-BSM, on the other hand, will yield a much worse performance with larger data deviations. This observation further showcases **CADCA**'s advantage in that its identification of anomalous data does not directly rely on data causality. A similar pattern can also be observed in T2–T4. As expected, **CADCA**

Δd	T1				T2				T3				T4			
	CADCA SR	R-B SR	**CADCA** IBR	R-B IBR	**CADCA** SR	R-B SR	**CADCA** IBR	R-B IBR	**CADCA** SR	R-B SR	**CADCA** IBR	R-B IBR	**CADCA** SR	R-B SR	**CADCA** IBR	R-B IBR
Object Detection Rate = 0.1																
6	90.43	87.83	0.00	0.85	88.89	44.44	1.12	0.00	100	0.00	10.53	0.00	100	100	1.56	1.56
7	90.43	78.26	0.00	3.83	88.89	27.78	1.12	0.00	100	0.00	10.53	0.00	100	60.00	1.56	1.56
8	90.43	76.52	0.00	2.98	88.89	22.22	0.56	0.00	100	0.00	10.53	0.00	100	0.00	0.00	0.00
9	90.43	66.09	0.00	2.55	72.22	22.22	0.56	0.00	100	0.00	10.53	0.00	100	0.00	0.00	0.00
10	90.43	60.00	0.00	6.38	94.44	22.22	1.12	0.00	100	0.00	10.53	0.00	100	0.00	0.00	0.00
Object Detection Rate = 0.5																
6	98.25	68.42	2.44	0.73	50.00	50.00	0.00	0.00	95.12	0.00	0.00	0.00	100	100	1.56	1.56
7	98.25	57.89	2.44	0.73	50.00	33.33	0.00	0.00	95.12	0.00	0.00	0.00	100	60.00	1.56	1.56
8	98.25	47.37	2.44	0.73	66.67	27.78	0.00	0.00	95.12	0.00	0.00	0.00	100	0.00	0.00	0.00
9	98.25	35.09	2.44	0.73	66.67	22.22	0.56	0.56	95.12	0.00	0.00	0.00	100	0.00	0.00	0.00
10	98.25	31.58	2.44	0.73	66.67	22.22	0.56	0.56	95.12	0.00	0.00	0.00	100	0.00	0.00	0.00
Object Detection Rate = 0.9																
6	98.33	68.33	1.96	0.49	50.00	50.00	0.00	0.00	97.50	0.00	0.00	0.00	100	100	1.56	1.56
7	98.33	53.33	1.96	0.25	55.56	33.33	0.00	0.00	97.50	0.00	0.00	0.00	100	60.00	1.56	1.56
8	98.33	41.67	1.96	0.49	66.67	27.78	0.00	0.00	97.50	0.00	0.00	0.00	100	0.00	0.00	0.00
9	98.33	31.67	1.96	0.74	66.67	22.22	0.56	0.56	97.50	0.00	0.00	0.00	100	0.00	0.00	0.00
10	98.33	28.33	1.96	0.98	66.67	22.22	0.00	0.00	97.50	0.00	0.00	0.00	100	0.00	0.00	0.00

Table 5.5: **CADCA** and RA-BSM's performance (%) under both sensor/algorithm failure and location manipulation.

shows a slightly worse performance in T2 due to the unpredictability of VSC's behavior. However, it can still achieve up to 72.22(= 94.44 - 22.22)% absolute SR increase over RA-BSM.

5.5.7 Computation Time Analysis

We evaluate **CADCA**'s computation time based on a 2016 MacBook Pro with 2.6 GHz Quad-Core Intel Core i7 CPU. Specifically, **CADCA** is implemented in Matlab with single-thread execution. We measure the execution time when there are 2–50 vehicles (including the ego vehicle). Fig. 5.9 shows the average computation time linearly increasing with the number of vehicles. This result matches our analysis in Chapter 5.4.3.8 that **CADCA**'s algorithms have $O(NK) \propto O(N)$ computation complexity when K, the number of data types, is fixed and N, the number of DSs, is proportional to the number of vehicles. Specifically, even if **CADCA** operates in the scenario with 50 vehicles, it only requires a 33.3ms execution time on average, which is only 0.7% of T_S (*i.e.*, the medium time to achieve a smooth control transition) and 1.2% of T_R (*i.e.*, the driver's reaction time in an urgent situation).

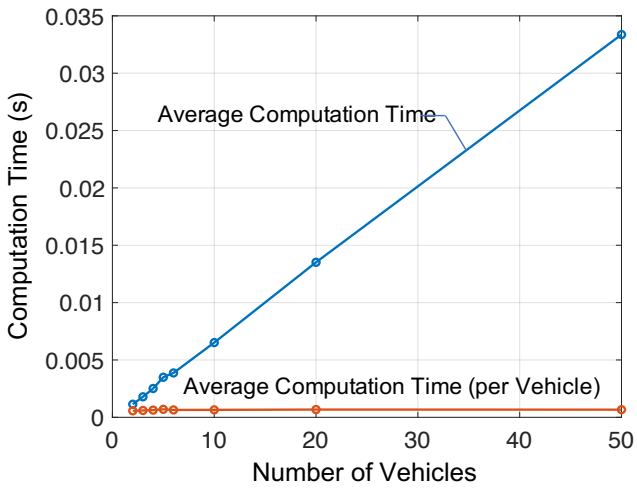


Figure 5.9: The computation time of **CADCA**. The execution time is computed based on the average of 10,000 executions under scenarios with the same number of vehicles.

5.5.8 Robustness Analysis

We now evaluate the robustness of **CADCA** in identifying anomalous entities as a stand-alone function to understand if and how an adversary can mislead **CADCA** (*i.e.*, incorrectly identifying entities as anomalous and lead to the identification of an incorrect probable context). Specifically, we control the output of **CADCA**'s consistency check to showcase its performance under different threat levels, assuming an attacker can always achieve his/her goal in manipulating data without any constraint. We tested all possible anomalous data combinations when there are up to 3 anomalous entities with $\leq$13 anomalous data when there can be up to 10 vehicles.

Fig. 5.10 shows the inclusion rates (IRs) and exact identification rates (EIRs) of anomalous entities under different attack/detection scenarios. IR is defined as the probability that **CADCA** include all anomalous entities in its output and EIR is defined as the probability that at least one algorithm introduced in Chapter 5.4.3 captures the exact anomalous entities without any false-positives or false-negatives. Specifically, Fig. 5.10 shows that **CADCA** can achieve 100% IRs and EIRs for an ideal/low-threat scenario of no false-positives/negatives in consistency checks. **CADCA** can also achieve $>$98% IRs even if all DSs can only achieve an 80–90% true-positive rate (TPR) and have a high (1–5%) false-positive rate (FPR). Note the EIRs in Figs. 5.10b and 5.10c show an increasing trend when there are less than 7 anomalous data types. This is the result of **CADCA** not being able to accurately pinpoint the exact anomalous entities because the high FPR in the consistency check phase will have more impacts on the local view construction if there are fewer true-positives among all positive detections. That is, **CADCA** will determine some normal entities as potentially anomalous based on its design. Nevertheless, **CADCA** can still achieve high IRs ($>$98%) as mentioned earlier.

We now consider mid/high-level attacks where the adversary can manipulate multiple DTs simultaneously to evade the consistency checks of certain DSs. Specifically, Fig. 5.10d shows **CADCA**'s performance under all possible multi-DT attacks and the TPR of a DS

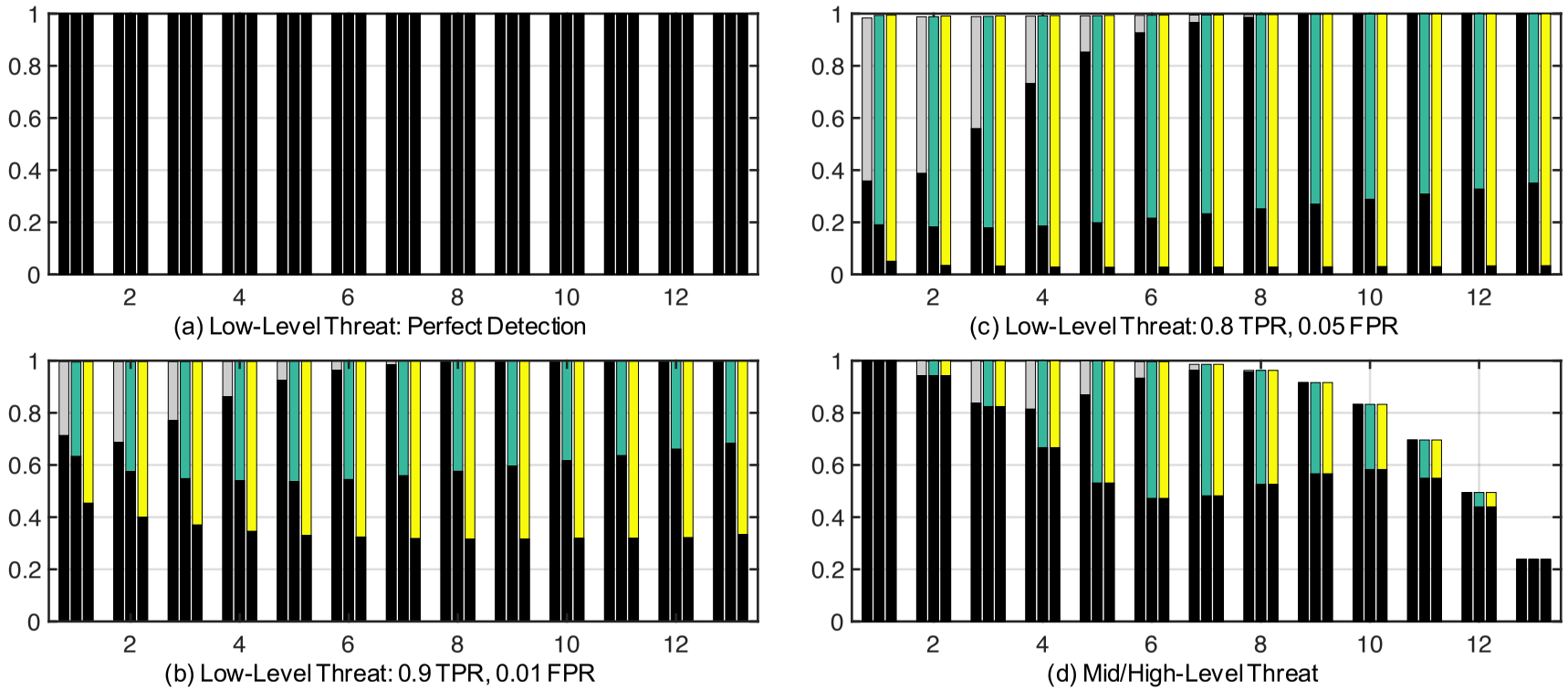


Figure 5.10: **CADCA**'s performance of identifying anomalous entities, where colored (black) bars represent **CADCA**'s IRs (EIRs), the x-axis in each figure represents the number of anomalous data (n), and the bars with the same n from left to right indicate the scenarios in which there are 3, 5, and 10 vehicles.

will be purposely set to 0% if all DTs in that DS are the targets of an attack. The results of EIRs may not seem to have an obvious trend/pattern when the number of anomalous DTs increases because the DSs considered in **CADCA** cannot exhaustively cover all possible DT combinations. Therefore, the EIRs under mid/high-level threat will be influenced by the design and the (imbalanced) coverage of the DSs w.r.t. to different data types. However, **CADCA** can still achieve a good and stable (*i.e.*, with a gradually decreasing trend) IR performance thanks to its design to report all potentially anomalous entities based on the output of local view construction. Specifically, even when two thirds of the data in all anomalous entities are coordinately manipulated (*i.e.*, the group of 12 anomalous data in Fig. 5.10d), **CADCA** can still achieve $\sim$50% IR.

5.6 Discussion

Limitations and Future Work. As a natural limitation in the absence of trusted data, **CADCA** cannot provide correct control decisions if *all or majority (two-thirds of data in anomalous entities)* of the received/perceived data have been manipulated to match their normal correlation. This will cause **CADCA** to identify the anomalous entities incorrectly or determine all the data to be potentially anomalous, depending on the data manipulation.

Also, **CADCA** is less effective when the maneuver of non-ego vehicles drastically changes during the attack as T2 and T4 shown in Chapters 5.5.5 and 5.5.6, and the current design of **CADCA** will only choose the control with the largest TTC when all control inputs are determined to be unsafe without recommending an alternative to avoid collision. Integration of traffic/maneuver patterns for behavior prediction and control recommendation is thus part of our future work.

Practical Deployment. While **CADCA** can also have incorrect blocking of inputs/controls, imperfect implementation and faults, engineers should implement a user-interface to notify the driver whenever a control/input is blocked, and also design a fail-safe mechanism to unblock the manual control of the vehicle, such as a panic button. However, this

fail-safe mechanism must ensure the central control/management center[4] will be notified and can reactivate **CADCA** or take over the control of the vehicle, if necessary, to prevent malicious drivers from abusing this fail-safe mechanism.

5.7 Conclusions

To reduce/eliminate the danger of static assignment of control priority in safety-critical situations due to potential component failures/attacks and malicious driver's input, we have proposed **CADCA**, a control decision-maker for semi-autonomous systems, that can perform risk assessment and resolve control conflicts when any of the received/perceived data is anomalous. Specifically, **CADCA** selects a control input that is safe to execute under uncertain situations. Our extensive evaluation has shown **CADCA** to achieve a 98% success rate in avoiding use of unsafe control inputs (T1) and have a $\geq$0.4 more success rate than the latest representative prior work for most commonly seen scenarios (T2).

[4]The deployment of central control/management center is expected to be commonly used for future autonomous vehicles and fleets [10, 38, 110, 131].

CHAPTER VI

Conclusions and Future Directions

6.1 Conclusions

This thesis has demonstrated the feasibility of enhancing the robustness and safety of SA systems by cross-validating the received data and the operation contexts. Specifically, we proposed **SafeSA**, a framework that consists of four subsystems — **CADD**, **DiVa**, **EDRoad**, and **CADCA** — for the integrity verification of system operation, received information, and control decisions.

6.1.1 Integrity Verification of System Operation (CADD and DiVa)

The adoption of electronic components introduces new vulnerabilities to modern SA systems. To cope with this double-edged trend, we proposed **CADD** (Chapter II) to determine whether or not the system operation matches its control inputs. Specifically, while the dynamics of SA systems depends heavily on its operation contexts and the sensors in SA systems cannot always capture the correct information, **CADD** utilizes multiple data groups to perform anomaly detection by cross-validating the estimations of the operation contexts and narrow down the search space for anomalous groups or components. That way, **CADD** i) eliminates the requirement of knowing the correct context before performing anomaly detection, ii) takes the operation context into account, and iii) does not require a fixed set of trusted sources for its detection.

After an anomaly is detected, **DiVa** (Chapter III) is proposed to further identify the data or components that have caused the abnormal system behavior. **DiVa** constructs hierarchical *detection sets* (DSs), *i.e.*, groups of data that describe certain correlation or causality of the target SA system, to cross-validate the input data. **DiVa** is able to perform consistency checks based on the DSs and formulate a Boolean equation system whose solution indicates which data are potentially anomalous, in order to identify the anomaly source(s) even when no data can be trusted entirely.

6.1.2 Integrity Verification of Received Information (EDRoad)

While future SA systems are expected to receive support from infrastructures and/or other SA entities, using incorrect information in services designed to support SA systems can degrade/risk their operation efficiency and security/safety. To address this critical problem, we proposed **EDRoad** (Chapter IV), a novel system and an ensemble learning framework for verifying the integrity of *individual* data segments/types received from external SA entities and restoring the identified anomalous data. **EDRoad**'s design can significantly reduce engineers' development effort, *at design time*, when setting up data verification schemes under different application requirements.

6.1.3 Integrity Verification of Control Decision (CADCA)

Since neither autonomous nor manual control is perfect, a disagreement between them may arise. To resolve such a control conflict, safety/mission-critical features in modern SA systems are usually implemented with *static priority assignment*. However, this static priority assignment can lead to catastrophic accidents when accompanied with faulty/compromised sensor readings (*e.g.*, the crashes of Boeing 737 MAX). To mitigate the grave danger of utilizing incorrect data for generating control decisions and learn from the incidents/crashes of Boeing 737 MAX, we proposed **CADCA** (Chapter V), a novel control decision-maker for SA systems, that detects both data manipulations and malicious/erro-

neous control inputs with the ultimate goal of resolving conflicting control inputs to ensure the system safety. **CADCA** utilizes the received data to establish an equation system for identifying the probable operation scenarios and performs risk assessments for all potential ⟨control input, context⟩ scenarios to recommend a final (and safe) control decision.

6.2 Future Directions

Finally, we discuss three potential research directions that can be extended from this thesis.

6.2.1 Identification of Context Correlation

While engineers must provide domain-specific formulations as inputs to **SafeSA** for its real-world implementation, we expect this prerequisite can also be replaced by an automatic training process. However, according to our evaluation in Chapters II to IV, there still does not exist any effective solution to automatically capture the correlation between system behavior and the operation context. Specifically, even **EDRoad** requires some manual configuration to incorporate domain-specific knowledge into the detection system, but it is a step forward in this direction. This training mechanism is expected to significantly reduce engineers' efforts in identifying new causal correlations between advanced sensors and components that have yet to be integrated in modern SA systems.

6.2.2 Generation of Alternative Control and Limp Mode Support

In the current design of **CADCA**, it will only delay the occurrence of an unsafe system state if all the control inputs are determined to be unsafe. A natural extension of **CADCA** is to explore how to derive a safe control when no control inputs are safe to execute. In an ideal scenario, this safe control should either resolve the safety-critical situation or make sure that the SA system will not enter any unsafe state in the foreseeable future and can keep operating until maintenance or repairment is available (*i.e.*, the limp mode).

6.2.3 SA Entity Cooperation

As (the current design of) **SafeSA** only assumes to passively receive information from other SA entities, it is also worth exploring if there is a feasible approach to have the ego SA system to *actively* request certain data from other entities while providing a sanity check to ensure the received information is correct without the need of their direct measurements as assumed in **EDRoad**. For example, an SA vehicle may request the camera image captured by another SA vehicle one block ahead and make sure that the received image is correct and accurate for its future path planning. This functionality can enable various future services and applications in SA systems.

BIBLIOGRAPHY

BIBLIOGRAPHY

[1] Amr Abdelaziz, Ron Burton, Frank Barickman, John Martin, Josh Weston, and Can Emre Koksal. Enhanced authentication based on angle of signal arrivals. *IEEE Transactions on Vehicular Technology*, 68(5):4602–4614, 2019.

[2] Alice Agogino, Kai Goebel, and Satnam Alag. Intelligent Sensor Validation and Sensor Fusion for Reliability and Safety Enhancement in Vehicle Control. In *UC Berkeley: California Partners for Advanced Transportation Technology*. UC Berkeley, 1995.

[3] Omar Y. Al-Jarrah, Carsten Maple, Mehrdad Dianati, David Oxtoby, and Alex Mouzakitis. Intrusion Detection Systems for Intra-Vehicle Networks: A Review, 2019.

[4] Saif Al-Sultan, Ali H. Al-Bayatti, and Hussein Zedan. Context-aware driver behavior detection system in intelligent transportation systems. *IEEE Transactions on Vehicular Technology*, 62(9):4264–4275, 2013.

[5] Malintha Amarasinghe, Sasikala Kottegoda, Asiri Liyana Arachchi, Shashika Muramudalige, H. M. N. Dilum Bandara, and Afkham Azeez. Cloud-based driver monitoring and vehicle diagnostic with OBD2 telematics. In *2015 IEEE International Conference on Electro/Information Technology (EIT)*, pages 505–510. IEEE, May 2015.

[6] American Association of State Highway and Transportation Officials. *A policy on geometric design of highways and streets, 2018.* American Association of State Highway and Transportation Officials, 2018.

[7] Wissam Aoudi, Mikel Iturbe, and Magnus Almgren. Truth will out: Departure-based process-level detection of stealthy attacks on control systems. In *Proceedings of the ACM Conference on Computer and Communications Security*, pages 817–831, New York, NY, USA, Oct 2018. Association for Computing Machinery.

[8] Shai A. Arogeti, Danwei Wang, Chang Boon Low, and Ming Yu. Fault detection isolation and estimation in a vehicle steering system. *IEEE Transactions on Industrial Electronics*, 59(12):4810–4820, 2012.

[9] Minjin Baek, Donggi Jeong, Dongho Choi, and Sangsun Lee. Vehicle Trajectory Prediction and Collision Warning via Fusion of Multisensors and Wireless Vehicular Communications. *Sensors (Basel, Switzerland)*, 20(1), Jan 2020.

[10] Riley Beggin and Kalea Hall. Autonomous Vehicle Industry Faces Inconsistent Guidelines. https://www.govtech.com/policy/autonomous-vehicle-industry-faces-inconsistent-guidelines.html. Archived: 2022-05-03, (https://web.archive.org/web/20220503160421/https://www.govtech.com/policy/autonomous-vehicle-industry-faces-inconsistent-guidelines.html).

[11] Adam Berthelot, Andreas Tamke, Thao Dang, and Gabi Breuel. Handling uncertainties in criticality assessment. In *IEEE Intelligent Vehicles Symposium, Proceedings*, pages 571–576, 2011.

[12] Christopher M. Bishop. *Pattern recognition and machine learning*. Springer, 2006.

[13] Norbert Bismeyer, Sebastian Mauthofer, Kpatcha M. Bayarou, and Frank Kargl. Assessment of node trustworthiness in VANETs using data plausibility checks with particle filters. In *IEEE Vehicular Networking Conference, VNC*, pages 78–85, 2012.

[14] Boeing. Boeing 737 MAX Updates – 737 MAX Return To Service Updates & Information. https://www.boeing.com/737-max-updates. Archived: 2022-05-03, (https://web.archive.org/web/20220422151901/https://www.boeing.com/737-max-updates/).

[15] Safa Boumiza and Rafik Braham. Intrusion threats and security solutions for autonomous vehicle networks. In *Proceedings of IEEE/ACS International Conference on Computer Systems and Applications, AICCSA*, volume 2017-October, pages 120–127. IEEE Computer Society, Mar 2018.

[16] Mattias Brännström, Erik Coelingh, and Jonas Sjöberg. Model-based threat assessment for avoiding arbitrary vehicle collisions. *IEEE Transactions on Intelligent Transportation Systems*, 11(3):658–669, Sep 2010.

[17] John L. Campbell, James L. Brown, Justin S. Graving, Monica G. Lichty, Thomas Sanquist, Diane N. Williams, and Justin F. Morgan. Human Factors Design Guidance For Driver-Vehicle Interfaces. Technical report, U.S. Department of Transportation, National Highway Traffic Safety Administration, 2016.

[18] Jie Chen, Ron J. Patton, and Hong Yue Zhang. Design of unknown input observers and robust fault detection filters. *International Journal of Control*, 63(1):85–105, Jan 1996.

[19] Qi Alfred Chen, Yucheng Yin, Yiheng Feng, Z Morley Mao, and Henry X Liu. Exposing Congestion Attack on Emerging Connected Vehicle based Traffic Signal Control. In *Network and Distributed Systems Security (NDSS) Symposium*, 2018.

[20] Rishu Chhabra, Seema Verma, and C. Rama Krishna. A survey on driver behavior detection techniques for intelligent transportation systems. In *Proceedings of the 7th International Conference Confluence 2017 on Cloud Computing, Data Science and Engineering*, pages 36–41. Institute of Electrical and Electronics Engineers Inc., Jun 2017.

[21] Rishu Chhabra, Seema Verma, and C. Rama Krishna. Detecting Aggressive Driving Behavior using Mobile Smartphone. In *Lecture Notes in Networks and Systems*, volume 46, pages 513–521. Springer, 2019.

[22] Kyong-Tak Cho and Kang G. Shin. Fingerprinting Electronic Control Units for Vehicle Intrusion Detection. *USENIX Security Symposium*, 2016.

[23] Kyong-Tak Cho, Kang G. Shin, and Taejoon Park. CPS approach to checking norm operation of a brake-by-wire system. In *Proceedings of the ACM/IEEE Sixth International Conference on Cyber-Physical Systems - ICCPS '15*, pages 41–50, New York, New York, USA, 2015. ACM Press.

[24] Hongjun Choi, Wen-Chuan Lee, Yousra Aafer, Fan Fei, Zhan Tu, Xiangyu Zhang, Dongyan Xu, and Xinyan Deng. Detecting attacks against robotic vehicles: A control invariant approach. In *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security (CCS'18)*, pages 801–816, 2018.

[25] Hongjun Choi, Wen Chuan Lee, Yousra Aafer, Fan Fei, Zhan Tu, Xiangyu Zhang, Dongyan Xu, and Xinyan Deng. Detecting attacks against robotic vehicles: A control invariant approach. In *Proceedings of the ACM Conference on Computer and Communications Security*, pages 801–816, New York, NY, USA, Oct 2018. Association for Computing Machinery.

[26] Comma AI. GitHub - commaai/opendbc: democratize access to car decoder rings. `https://github.com/commaai/opendbc`. Archived: 2022-05-03, (`https://web.archive.org/web/20220503150437/https://github.com/commaai/opendbc`).

[27] Comma AI. Comma AI Open Dataset. `https://github.com/commaai/research`, 2016. Archived: 2022-05-03, (`https://web.archive.org/web/20220503142117/https://github.com/commaai/research`).

[28] Pritam Dash, Guanpeng Li, Zitao Chen, Mehdi Karimibiuki, and Karthik Pattabiraman. PID-Piper: Recovering Robotic Vehicles from Physical Attacks. In *IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)*, 2021.

[29] Robert I. Davis, Alan Burns, Victor Pollex, and Frank Slomka. On priority assignment for controller area network when some message identifiers are fixed. In *ACM International Conference Proceeding Series*, volume 04-06-November-2015, pages 279–288, New York, New York, USA, Nov 2015. Association for Computing Machinery.

[30] Johan de Kleer, Alan K. Mackworth, and Raymond Reiter. Characterizing diagnoses and systems. *Artificial Intelligence*, 56(2-3):197–222, 1992.

[31] Johan de Kleer and Brian C. Williams. Diagnosing multiple faults. *Artificial Intelligence*, 32(1):97–130, 1987.

[32] Stefan Dietzel, Rens Van Der Heijden, Hendrik Decke, and Frank Kargl. A flexible, subjective logic-based framework for misbehavior detection in V2V networks. In *Proceeding of IEEE International Symposium on a World of Wireless, Mobile and Multimedia Networks 2014, WoWMoM 2014*. Institute of Electrical and Electronics Engineers Inc., Oct 2014.

[33] Khalid L. A. El-Ashmawy. Investigation of the Accuracy of Google Earth Elevation Data. *Artificial Satellites*, 51(3), 2016.

[34] FAAC. Vehicle Simulation - Automotive Vehicle Simulation Software and Research. https://www.faac.com/realtime-technologies/solutions/automotive-vehicle-simulation-research/?gclid=CjwKCAjw49qKBhAoEiwAHQVTo20ekN2w6RCRIlVcemAXfZVLe3kkP7yNZkhHPw51VjGD9vHkSFFhbBoCIqIQAvD_BwE. Archived: 2022-05-03, (https://web.archive.org/web/20220503160111/https://www.faac.com/realtime-technologies/solutions/automotive-vehicle-simulation-research/).

[35] Zhihan Fang, Guang Wang, Xiaoyang Xie, Fan Zhang, and Desheng Zhang. Urban Map Inference by Pervasive Vehicular Sensing Systems with Complementary Mobility. *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies*, Mar 2021.

[36] Zhihan Fang, Yu Yang, Shuai Wang, Boyang Fu, Zixing Song, Fan Zhang, and Desheng Zhang. MAC: Measuring the Impacts of Anomalies on Travel Time of Multiple Transportation Systems. *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies*, Jun 2019.

[37] Federal Highway Administration. Safety Pilot Model Deployment Data - Data.gov. https://catalog.data.gov/dataset/safety-pilot-model-deployment-data. Archived: 2022-05-03, (https://web.archive.org/web/20220503150040/https://catalog.data.gov/dataset/safety-pilot-model-deployment-data).

[38] Johannes Feiler, Simon Hoffmann, and Frank Diermeyer. Concept of a Control Center for an Automated Vehicle Fleet. *2020 IEEE 23rd International Conference on Intelligent Transportation Systems, ITSC 2020*, Sep 2020.

[39] Cheng Feng, Venkata Reddy Palleti, Aditya Mathur, and Deeph Chana. A Systematic Framework to Generate Invariants for Anomaly Detection in Industrial Control Systems. In *Network and Distributed Systems Security (NDSS) Symposium 2019*, 2019.

[40] Ford Motor Company. Ford Accelerates Connectivity Strategy in China; Targets Production of First C-V2X-Equipped Vehicle in 2021. https://media.ford.com/content/fordmedia/fap/cn/en/news/2019/03/26/Ford_Accelerates_Connectivity_Strategy_in_China_And_Targets_Production_of_First_C-V2X-Equipped_Vehicle_in_2021.html. Archived:

2022-05-03, (https://web.archive.org/web/20220503155548/https://media.ford.com/content/fordmedia/fap/cn/en/news/2019/03/26/Ford_Accelerates_Connectivity_Strategy_in_China_And_Targets_Production_of_First_C-V2X-Equipped_Vehicle_in_2021.html).

[41] Thierry Fraichard and Hajime Asama. Inevitable collision states - a step towards safer robots ? Thierry Fraichard , Hajime Asama To cite this version : Inevitable Collision States. *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems October*, 18(10):2–9, 2003.

[42] Yoav Freund and Robert E. Schapire. A Decision-Theoretic Generalization of On-Line Learning and an Application to Boosting. *Journal of Computer and System Sciences*, Aug 1997.

[43] Jerome Friedman, Trevor Hastie, and Robert Tibshirani. Additive logistic regression: a statistical view of boosting (With discussion and a rejoinder by the authors). *The Annals of Statistics*, 28(2):337 – 407, 2000.

[44] Zhouyu Fu, Weiming Hu, and Tieniu Tan. Similarity based vehicle trajectory clustering and anomaly detection. In *Proceedings - International Conference on Image Processing, ICIP*, volume 2, pages 602–605, 2005.

[45] Arun Ganesan, Jayanthi Rao, and Kang Shin. Exploiting Consistency Among Heterogeneous Sensors for Vehicle Anomaly Detection. In *SAE*, Mar 2017.

[46] Zhiwei Gao, Carlo Cecati, and Steven X. Ding. A survey of fault diagnosis and fault-tolerant techniques-part I: Fault diagnosis with model-based and signal-based approaches. *IEEE Transactions on Industrial Electronics*, 62(6):3757–3767, Jun 2015.

[47] Zhiwei Gao, Steven X. Ding, and Yan Ma. Robust fault estimation approach and its application in vehicle lateral dynamic systems. *Optimal Control Applications and Methods*, 28(3):143–156, May 2007.

[48] Dominic Gates and Mike Baker. The inside story of mcas: How boeing's 737 max system gained power and lost safeguards. https://www.seattletimes.com/seattle-news/times-watchdog/the-inside-story-of-mcas-how-boeings-737-max-system-gained-power-and-lost-safeguards/. Archived: 2022-05-03, (https://web.archive.org/web/20220503144703/https://www.seattletimes.com/seattle-news/times-watchdog/the-inside-story-of-mcas-how-boeings-737-max-system-gained-power-and-lost-safeguards/).

[49] General Motors. Buick Debuts V2X Technology and Launches Refreshed GL6 MPV in China . https://media.gm.com/media/cn/en/gm/news.detail.html/content/Pages/news/cn/en/2020/Nov/1120-Buick.html. Archived: 2022-05-03, (https://web.archive.org/web/20220503155624/https://media.gm.com/media/cn/en/gm/news.detail.html/content/Pages/news/cn/en/2020/Nov/1120-Buick.html).

[50] J. Gerler, M. Costin, Xiaowen Fang, Z. Kowalczuk, M. Kunwer, and R. Monajemy. Model based diagnosis for automotive engines-algorithm development and testing on a production vehicle. *IEEE Transactions on Control Systems Technology*, 3(1):61–69, Mar 1995.

[51] GMC Division of General Motors. Explore GMC Safety & Driver Assistance Technology. `https://www.gmc.com/safety-features`. Archived: 2022-05-03, (`https://web.archive.org/web/20220503155236/https://www.gmc.com/safety-features`).

[52] GPS.gov. Official U.S. government information about the Global Positioning System (GPS) and related topics. `https://www.gps.gov/systems/gps/performance/accuracy/`. Archived: 2022-05-03, (`https://web.archive.org/web/20220502201844/https://www.gps.gov/systems/gps/performance/accuracy/`).

[53] Havard Fjaer Grip, Lars Imsland, Tor A. Johansen, Jens C. Kalkkuhl, and Avshalom Suissa. Estimation of road inclination and bank angle in automotive vehicles. In *2009 American Control Conference*, pages 426–432. IEEE, 2009.

[54] Fei Guo, Zichang Wang, Suguo Du, Huaxin Li, Haojin Zhu, Qingqi Pei, Zhenfu Cao, and Jianhong Zhao. Detecting Vehicle Anomaly in the Edge via Sensor Consistency and Frequency Characteristic. In *IEEE Transactions on Vehicular Technology*, pages 5618–5628. Institute of Electrical and Electronics Engineers Inc., Jun 2019.

[55] Pinyao Guo, Hunmin Kim, Nurali Virani, Jun Xu, Minghui Zhu, and Peng Liu. RoboADS: Anomaly Detection Against Sensor and Actuator Misbehaviors in Mobile Robots. In *2018 48th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)*, pages 574–585. IEEE, Jun 2018.

[56] Abhishek Gupta, Alagan Anpalagan, Ling Guan, and Ahmed Shaharyar Khwaja. Deep learning for object detection and scene perception in self-driving cars: Survey, challenges, and open issues. *Array*, 10:100057, 2021.

[57] David Halliday, Robert Resnick, and Jearl Walker. *Fundamentals of Physics*. Fundamentals of Physics. Wiley, 2013.

[58] Trevor Hastie, Robert Tibshirani, and Jerome Friedman. *The Elements of Statistical Learning – Data Mining, Inference, and Prediction, Second Edition*. arXiv, 2009.

[59] Tin Kam Ho. Random decision forests. *Proceedings of the International Conference on Document Analysis and Recognition, ICDAR*, 1:278–282, 1995.

[60] Jie Hu, Li Xu, Xin He, and Wuqiang Meng. Abnormal Driving Detection Based on Normalized Driving Behavior. *IEEE Transactions on Vehicular Technology*, 66(8):6645–6652, Aug 2017.

[61] ISO. CAN-FD - ISO 11898-1:2015. https://www.iso.org/standard/63648.html. Archived: 2022-05-03, (https://web.archive.org/web/20220503150302/https://www.iso.org/standard/63648.html).

[62] ISO. High Speed CAN - ISO 11898-2:2016. https://www.iso.org/standard/67244.html. Archived: 2022-05-03, (https://web.archive.org/web/20220503150345/https://www.iso.org/standard/67244.html).

[63] Attila Jaeger, Norbert Bißmeyer, Hagen Stübing, and Sorin A. Huss. A Novel Framework for Efficient Mobility Data Verification in Vehicular Ad-hoc Networks. *International Journal of Intelligent Transportation Systems Research*, 10(1):11–21, Jan 2012.

[64] Marek Jareslaw Jastrzebski, Michael John Cullen, Eric Blaine Ferch, and Stephen Alan De La Salle. System and method for torque based vehicle speed control, Aug 1997.

[65] Jens Jauch, Johannes Masino, Tim Staiger, and Frank Gauterin. Road Grade Estimation With Vehicle-Based Inertial Measurement Unit and Orientation Filter. *IEEE Sensors Journal*, 18(2):781–789, Jan 2018.

[66] Xiaomeng Ju and Matías Salibián-Barrera. Robust boosting for regression problems. *Computational Statistics & Data Analysis*, 153:107065, Jan 2021.

[67] Nico Kaempchen, Bruno Schiele, and Klaus Dietmayer. Situation assessment of an autonomous emergency brake for arbitrary vehicle-to-vehicle collision scenarios. *IEEE Transactions on Intelligent Transportation Systems*, 10(4):678–687, Dec 2009.

[68] Desmond King-Hele. Erasmus Darwin's Improved Design for Steering Carriages and Cars. *Notes and Records of the Royal Society of London*, 56(1), Jan 2002.

[69] Matthijs Klomp, Yunlong Gao, and Fredrik Bruzelius. Longitudinal velocity and road slope estimation in hybrid electric vehicles employing early detection of excessive wheel slip. *Vehicle System Dynamics*, 52:172–188, May 2014.

[70] Karl Koscher, Alexei Czeskis, Franziska Roesner, Shwetak Patel, Tadayoshi Kohno, Stephen Checkoway, Damon McCoy, Brian Kantor, Danny Anderson, Hovav Shacham, and Stefan Savage. Experimental Security Analysis of a Modern Automobile. In *2010 IEEE Symposium on Security and Privacy*, pages 447–462. IEEE, 2010.

[71] Omurcan Kumtepe, Gozde Bozdai Akar, and Enes Yuncu. On vehicle aggressive driving behavior detection using visual information. In *2015 23rd Signal Processing and Communications Applications Conference, SIU 2015 - Proceedings*, pages 795–798. Institute of Electrical and Electronics Engineers Inc., Jun 2015.

[72] Javier Kypuros. *System Dynamics and Control with Bond Graph Modeling*. Taylor & Francis, 2013.

[73] Andreas Lawitzky, Daniel Althoff, Christoph F. Passenberg, Georg Tanzmeister, Dirk Wollherr, and Martin Buss. Interactive scene prediction for automotive applications. In *IEEE Intelligent Vehicles Symposium, Proceedings*, pages 1028–1033, 2013.

[74] Jooyoung Lee and Kitae Jang. A framework for evaluating aggressive driving behaviors based on in-vehicle driving records. *Transportation Research Part F: Traffic Psychology and Behaviour*, 65:610–619, Aug 2019.

[75] Stéphanie Lefèvre, Dizan Vasquez, and Christian Laugier. A survey on motion prediction and risk assessment for intelligent vehicles, Dec 2014.

[76] Fu Li, Hai Zhang, Huan Che, and Xiaochen Qiu. Dangerous driving behavior detection using smartphone sensors. In *IEEE Conference on Intelligent Transportation Systems, Proceedings, ITSC*, pages 1902–1907. Institute of Electrical and Electronics Engineers Inc., Dec 2016.

[77] Luyang Liu, Hongyu Li, Jian Liu, Cagdas Karatas, Yan Wang, Marco Gruteser, Yingying Chen, and Richard P. Martin. BigRoad: Scaling Road Data Acquisition for Dependable Self-Driving. In *Proceedings of the 15th Annual International Conference on Mobile Systems, Applications, and Services - MobiSys '17*, pages 371–384, New York, New York, USA, 2017. ACM Press.

[78] Nai Wei Lo and Hsiao Chien Tsai. Illusion attack on VANET applications - A message plausibility problem. In *GLOBECOM - IEEE Global Telecommunications Conference*, 2007.

[79] Jianhui Luo, Fang Tu, Mohammad S. Azam, Krishna R. Pattipati, Peter K. Willett, Liu Qiao, and Masayuki Kawamoto. Intelligent model-based diagnostics for vehicle health management. In Peter K. Willett and Thiagalingam Kirubarajan, editors, *Proceedings Volume 5107, System Diagnosis and Prognosis: Security and Condition Monitoring Issues III*, volume 5107, pages 13–26. International Society for Optics and Photonics, Aug 2003.

[80] Yongfeng Ma, Ziyu Zhang, Shuyan Chen, Yanan Yu, and Kun Tang. A Comparative Study of Aggressive Driving Behavior Recognition Algorithms Based on Vehicle Motion Data. *IEEE Access*, 7:8028–8038, 2019.

[81] Muhammad Nasiruddin Mahyuddin, Jing Na, Guido Herrmann, Xuemei Ren, and Phil Barber. Adaptive Observer-Based Parameter Estimation With Application to Road Gradient and Vehicle Mass Estimation. *IEEE Transactions on Industrial Electronics*, 61(6):2851–2863, Jun 2014.

[82] Stephen Mangan and Jihong Wang. Development of a Novel Sensorless Longitudinal Road Gradient Estimation Method Based on Vehicle CAN Bus Data. *IEEE/ASME Transactions on Mechatronics*, 12(3):375–386, Jun 2007.

[83] Kirsten Matheus and Thomas Königseder. *Automotive Ethernet*. Cambridge University Press, 2014.

[84] MathWorks. Automated Driving Toolbox. https://www.mathworks.com/products/automated-driving.html/. Archived: 2022-05-03, (https://web.archive.org/web/20220503160025/https://www.mathworks.com/products/automated-driving.html.html).

[85] Mathworks. Genetic Algorithm - MATLAB. https://www.mathworks.com/help/gads/ga.html;jsessionid=3044f92e268c4e36cf7375a2f451. Archived: 2022-05-03, (https://web.archive.org/web/20220503161133/https://www.mathworks.com/help/gads/ga.html;jsessionid=3044f92e268c4e36cf7375a2f451).

[86] Russell B. Millar. *Maximum Likelihood Estimation and Inference: With Examples in R, SAS and ADMB*. Wiley, Jul 2011.

[87] Charlie Miller and Chris Valasek. Remote Exploitation of an Unaltered Passenger Vehicle. In *Black Hat USA 2015*, 2015.

[88] Charlie Miller and Chris Valasek. Advanced CAN Injection Techniques for Vehicle Networks. In *Presentation at BlackHat USA*, 2016.

[89] Alexandros Mouzakitis. Classification of fault diagnosis methods for control systems. *Measurement and Control (United Kingdom)*, 46(10):303–308, Dec 2013.

[90] Steven E. Muldoon, Mark Kowalczyk, and John Shen. Vehicle fault diagnostics using a sensor fusion approach. In *Proceedings of IEEE Sensors*, volume 2, pages 1591–1596. IEEE, 2002.

[91] Steffen Muller, Michael Uchanski, and Karl Hedrick. Estimation of the Maximum Tire-Road Friction Coefficient. *Journal of Dynamic Systems, Measurement, and Control*, 125(4):607–617, Jan 2004.

[92] Michael Muter and Naim Asaj. Entropy-based anomaly detection for in-vehicle networks. In *2011 IEEE Intelligent Vehicles Symposium (IV)*, pages 1110–1115. IEEE, Jun 2011.

[93] Michael Muter, Andre Groll, and Felix C. Freiling. A structured approach to anomaly detection for in-vehicle networks. In *2010 Sixth International Conference on Information Assurance and Security*, pages 92–98. IEEE, Aug 2010.

[94] National Highway Traffic Safety Administration, US Department of Transportation. Traffic Safety Facts. Technical report, US Department of Transportation, 2015.

[95] NHTSA. How does NHTSA categorize vehicles? (FAQ-06). https://www.nhtsa.gov/ratings. Archived: 2022-05-03, (https://web.archive.org/web/20220503151159/https://www.nhtsa.gov/ratings).

[96] Amy Nordrum. Smartphone Accelerometers Can Be Fooled by Sound Waves - IEEE Spectrum. https://spectrum.ieee.org/tech-talk/telecom/security/smartphone-accelerometers-can-be-fooled-by-sound-waves. Archived: 2022-05-03, (https://web.archive.org/web/20220503150946/https://spectrum.ieee.org/smartphone-accelerometers-can-be-fooled-by-sound-waves).

[97] Mattias Nyberg. A generalized minimal hitting-set algorithm to handle diagnosis with behavioral modes. *IEEE Transactions on Systems, Man, and Cybernetics Part A:Systems and Humans*, 41(1):137–148, Jan 2011.

[98] OpenXC. OpenXC Vehicle Interface Reference Design.

[99] R. J. Patton, J. Chen, and S. B. Nielsen. Model-based methods for fault diagnosis: Some guidelines. *Transactions of the Institute of Measurement & Control*, 17(2):73–83, 1995.

[100] Florian Pölzlbauer, Robert I. Davis, and Iain Bate. A practical message ID assignment policy for controller area network that maximizes extensibility. In *ACM International Conference Proceeding Series*, volume 19-21-October-2016, pages 45–54, New York, New York, USA, Oct 2016. Association for Computing Machinery.

[101] Andrzej Puchalski. A technique for the vibration signal analysis in vehicle diagnostics. *Mechanical Systems and Signal Processing*, 56-57:173–180, May 2015.

[102] Qiao Sun. Sensor fusion for vehicle health monitoring and degradation detection. In *Proceedings of the Fifth International Conference on Information Fusion. FUSION 2002. (IEEE Cat.No.02EX5997)*, volume 2, pages 1422–1427. Int. Soc. Inf. Fusion, 2002.

[103] Qualcomm. 5G NR-based C-V2X (Qualcomm). https://www.qualcomm.com/invention/5g/cellular-v2x. Archived: 2022-05-03, (https://web.archive.org/web/20220503155512/https://www.qualcomm.com/research/5g/cellular-v2x).

[104] Raul Quinonez, Jairo Giraldo, Luis Salazar, Santa Cruz, and Erick Bauman. SAVIOR: Securing Autonomous Vehicles with Robust Physical Invariants. In *29th USENIX Security Symposium*, 2020.

[105] Gopi Krishnan Rajbahadur, Andrew J. Malton, Andrew Walenstein, and Ahmed E. Hassan. A Survey of Anomaly Detection for Connected Vehicle Cybersecurity and Safety. In *IEEE Intelligent Vehicles Symposium, Proceedings*, volume 2018-June, pages 421–426. Institute of Electrical and Electronics Engineers Inc., Oct 2018.

[106] Economic Times. Simulation and Testing: This is how simulation and testing works for vehicle development, Auto News, ET Auto. https://auto.economictimes.indiatimes.com/news/auto-technology/this-is-how-simulation-testing-works-for-vehicle-development/64254482. Archived:

2022-05-03, (https://web.archive.org/web/20220503160201/https://auto.economictimes.indiatimes.com/news/auto-technology/this-is-how-simulation-testing-works-for-vehicle-development/64254482).

[107] Engineering360. Inclinometer. https://www.globalspec.com/industrial-directory/inclinometer. Archived: 2022-05-03, (https://web.archive.org/web/20220503154856/https://www.globalspec.com/industrial-directory/inclinometer).

[108] French air safety investigation authority for civil aviation. Accident to the airbus a320-211, registered d-aipx and operated by germanwings, flight gwi18g, on 03/24/15 at prads-haute-bléone. https://bea.aero/en/investigation-reports/notified-events/detail/accident-to-the-airbus-a320-211-registered-d-aipx-and-operated-by-germanwings-flight-gwi18g-on-03-24-15-at-prads-haute-bleone. Archived: 2022-05-03, (https://web.archive.org/web/20220503145844/https://bea.aero/en/investigation-reports/notified-events/detail/accident-to-the-airbus-a320-211-registered-d-aipx-and-operated-by-germanwings-flight-gwi18g-on-03-24-15-at-prads-haute-bleone).

[109] Mechanical Simulation. Carsim. https://www.carsim.com/products/carsim/index.php. Archived: 2022-05-03, (https://web.archive.org/web/20220503144113/https://www.carsim.com/products/carsim/index.php).

[110] T-Systems. Teleoperated driving controls cars remotely. https://www.t-systems.com/de/en/industries/automotive/connected-mobility/teleoperated-driving. Archived: 2022-05-03, (https://web.archive.org/web/20220503160634/https://www.t-systems.com/de/en/industries/automotive/connected-mobility/teleoperated-driving).

[111] TE Connectivity. Using Sensor Fusion to Improve the Performance of Tilt Sensors White Paper. https://www.te.com/usa-en/industries/sensor-solutions/insights/tilt-sensors-white-paper.html. Archived: 2022-05-03, (https://web.archive.org/web/20220503154716/https://www.te.com/usa-en/industries/sensor-solutions/insights/tilt-sensors-white-paper.html).

[112] Tilt-Tech CC. Tiltmeters and Inclinometers. https://tilt-tech.co.za/T-TNEW/products/tilt-meters-inclinometers/. Archived: 2022-05-03, (https://web.archive.org/web/20220503154811/https://tilt-tech.co.za/T-TNEW/products/tilt-meters-inclinometers/).

[113] Ishtiaq Rouf, Rob Miller, Hossen Mustafa, Travis Taylor, Sangho Oh, Wenyuan Xu, Marco Gruteser, Wade Trappe, and Ivan Seskar. Security and privacy vulnerabilities of in-car wireless networks: a tire pressure monitoring system case study. *Proceedings of the 19th USENIX conference on Security*, pages 21–21, 2010.

[114] SAE International. J2735D: Dedicated Short Range Communications (DSRC) Message Set Dictionary - SAE International. https://www.sae.org/standards/content/j2735_201603/. Archived: 2022-05-03, (https://web.archive.org/web/20220503151437/https://www.sae.org/standards/content/j2735_201603/).

[115] SAE International. SAE J1979A: E/E Diagnostic Test Modes. https://www.sae.org/standards/content/j1979_201702/. Archived: 2022-05-03, (https://web.archive.org/web/20220503150648/https://www.sae.org/standards/content/j1979_201702/).

[116] Fatih Sakiz and Sevil Sen. A survey of attacks and detection mechanisms on intelligent transportation systems: VANETs and IoV, Jun 2017.

[117] Ankur Sarker and Haiying Shen. A Data-Driven Misbehavior Detection System for Connected Autonomous Vehicles. *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies*, Dec 2018.

[118] Matthias Schäfer, Patrick Leu, Vincent Lenders, and Jens Schmitt. Secure motion verification using the doppler effect. In *Proceedings of the 9th ACM Conference on Security & Privacy in Wireless and Mobile Networks*, pages 135–145, 2016.

[119] Chris Seiffert, Taghi M. Khoshgoftaar, Jason Van Hulse, and Amri Napolitano. RUSBoost: A hybrid approach to alleviating class imbalance. *IEEE Transactions on Systems, Man, and Cybernetics Part A:Systems and Humans*, 40(1):185–197, Jan 2010.

[120] Junjie Shen, Jun Yeon Won, Zeyuan Chen, and Qi Alfred Chen. Drift with Devil: Security of Multi-Sensor Fusion based Localization in High-Level Autonomous Driving under GPS Spoofing (Extended Version). *arXiv*, Jun 2020.

[121] Bin Shi, Li Xu, Jie Hu, Yun Tang, Hong Jiang, Wuqiang Meng, and Hui Liu. Evaluating Driving Styles by Normalizing Driving Behavior Based on Personalized Driver Modeling. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 45(12):1502–1508, Dec 2015.

[122] Yasser Shoukry, Pierluigi Nuzzo, Alberto Puggelli, Alberto L. Sangiovanni-Vincentelli, Sanjit A. Seshia, and Paulo Tabuada. Secure State Estimation for Cyber-Physical Systems under Sensor Attacks: A Satisfiability Modulo Theory Approach. *IEEE Transactions on Automatic Control*, 62(10):4917–4932, Oct 2017.

[123] Gurdit Singh, Divya Bansal, and Sanjeev Sofat. A smartphone based technique to monitor driving behavior using DTW and crowdsensing. *Pervasive and Mobile Computing*, 40:56–70, Sep 2017.

[124] Isaac Skog and Peter Handel. Indirect Instantaneous Car-Fuel Consumption Measurements. *IEEE Transactions on Instrumentation and Measurement*, 63(12):3190–3198, Dec 2014.

[125] Jessy George Smith, Sai Kirthi Ponnuru, and Mandar Patil. Detection of aggressive driving behavior and fault behavior using pattern matching. In *2016 International Conference on Advances in Computing, Communications and Informatics, ICACCI 2016*, pages 207–211. Institute of Electrical and Electronics Engineers Inc., Nov 2016.

[126] Steven So, Jonathan Petit, and David Starobinski. Physical layer plausibility checks for misbehavior detection in v2x networks. In *Proceedings of the 12th Conference on Security and Privacy in Wireless and Mobile Networks*, pages 84–93, 2019.

[127] Hagen Stübing, Jonas Firl, and Sorin A Huss. A two-stage verification process for car-to-x mobility data based on path prediction and probabilistic maneuver recognition. In *2011 IEEE Vehicular Networking Conference (VNC)*, pages 17–24. IEEE, 2011.

[128] Beshr Sultan and Mike Mcdonald. Assessing The Safety Benefit of Automatic Collision Avoidance Systems (During Emergency Braking Situations). Technical report, University of Southampton, 2003.

[129] Mingshun Sun, Ming Li, and Ryan Gerdes. A data trust framework for VANETs enabling false data detection and secure vehicle tracking. In *2017 IEEE Conference on Communications and Network Security (CNS)*, volume 2017-Janua, pages 1–9. IEEE, Oct 2017.

[130] U.S. Geological Survey. U.S. Geological Survey - Global Positioning Application and Practice. `https://water.usgs.gov/osw/gps/`. Archived: 2022-05-03, (`https://web.archive.org/web/20220503144152/https://water.usgs.gov/osw/gps/`).

[131] Camil Tahan, Gerasimos Skaltsas, and Nabil Katicha. Enabling the mass adoption of autonomous driving. Technical report, Strategy & Middle East, part of the PwC network, 2020.

[132] Tesla. Tesla Model 3 . `https://www.tesla.com/model3`. Archived: 2022-05-03, (`https://web.archive.org/web/20220503150845/https://www.tesla.com/model3`).

[133] Timothy Trippel, Ofir Weisse, Wenyuan Xu, Peter Honeyman, and Kevin Fu. WALNUT: Waging Doubt on the Integrity of MEMS Accelerometers with Acoustic Injection Attacks. In *Proceedings - 2nd IEEE European Symposium on Security and Privacy, EuroS and P 2017*, pages 3–18. Institute of Electrical and Electronics Engineers Inc., Jun 2017.

[134] United States Department of Transportation. AERIS Operational Scenarios. `https://www.its.dot.gov/research_archives/aeris/pdf/AERIS_Operational_Scenarios011014.pdf`. Archived: 2022-05-03, (`https://web.archive.org/web/20220412140659/https://its.dot.gov/research_archives/aeris/pdf/AERIS_Operational_Scenarios011014.pdf`).

[135] United States Department of Transportation. Intelligent Transportation Systems - Connected Vehicle Pilot Deployment Program. https://www.its.dot.gov/pilots/. Archived: 2022-05-03, (https://web.archive.org/web/20220423082716/https://www.its.dot.gov/pilots/).

[136] United States Department of Transportation. Intelligent Transportation Systems - ITS in Use Today. https://www.its.dot.gov/resources/fastfacts.htm. Archived: 2022-05-03, (https://web.archive.org/web/20220503151327/https://www.its.dot.gov/resources/fastfacts.htm).

[137] David I. Urbina, Jairo Giraldo, Alvaro A. Cardenas, Nils Ole Tippenhauer, Junia Valente, Mustafa Faisal, Justin Ruths, Richard Candell, and Henrik Sandberg. Limiting the impact of stealthy attacks on Industrial Control Systems. In *Proceedings of the ACM Conference on Computer and Communications Security*, volume 24-28-October-2016, pages 1092–1105, New York, New York, USA, Oct 2016. Association for Computing Machinery.

[138] USEPA. Highlights of the Automotive Trends Report. https://www.epa.gov/automotive-trends/highlights-automotive-trends-report. Archived: 2022-05-03, (https://web.archive.org/web/20220503151122/https://www.epa.gov/automotive-trends/highlights-automotive-trends-report).

[139] Eser Ustunel and Engin Masazade. Vision-based road slope estimation methods using road lines or local features from instant images. *IET Intelligent Transport Systems*, 13(10), Oct 2019.

[140] Chris Valasek and Charlie Miller. Adventures in Automotive Networks and Control Units. In *DefCon21*, 2013.

[141] Rens Wouter van der Heijden, Stefan Dietzel, Tim Leinmuller, and Frank Kargl. Survey on Misbehavior Detection in Cooperative Intelligent Transportation Systems. *IEEE Communications Surveys & Tutorials*, 21(1):779–811, 2019.

[142] Liang Wang, Yihuan Zhang, and Jun Wang. Map-Based Localization Method for Autonomous Vehicles Using 3D-LIDAR. *IFAC-PapersOnLine*, 50(1):276–281, Jul 2017.

[143] Qiujie Wang, Tao Jin, and Mohamed A. Mohamed. An Innovative Minimum Hitting Set Algorithm for Model-Based Fault Diagnosis in Power Distribution Network. *IEEE Access*, 7:30683–30692, 2019.

[144] Armin Wasicek and Andre Weimerskirch. Recognizing Manipulated Electronic Control Units. In *SAE*, Apr 2015.

[145] Wufei Wu, Renfa Li, Guoqi Xie, Jiyao An, Yang Bai, Jia Zhou, and Keqin Li. A survey of intrusion detection for in-vehicle networks. *IEEE Transactions on Intelligent Transportation Systems*, 21(3):919–933, Mar 2020.

[146] Lu Xi, Xu Xiangyang, and Liu Yanfang. Simulation of Gear-shift Algorithm for Automatic Transmission Based on MATLAB. In *2009 WRI World Congress on Software Engineering*, pages 476–480. IEEE, 2009.

[147] Bin Xiao, Bo Yu, and Chuanshan Gao. Detection and localization of sybil nodes in vanets. In *Proceedings of the 2006 workshop on Dependability issues in wireless ad hoc networks and sensor networks (DIWANS'06)*, pages 1–8, 2006.

[148] XieXiaoyang, YangYu, FangZhihan, WangGuang, ZhangFan, ZhangFan, LiuYunhuai, and ZhangDesheng. coSense: Collaborative Urban-Scale Vehicle Sensing Based on Heterogeneous Fleets. *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies*, Dec 2018.

[149] Jie Xiong and Kyle Jamieson. Securearray: Improving wifi security with fine-grained physical-layer information. In *Proceedings of the 19th annual international conference on Mobile computing & networking (MobiCom'13)*, pages 441–452, 2013.

[150] Danfeng (Daphne) Yao, Xiaokui Shu, Long Cheng, and Salvatore J. Stolfo. Anomaly Detection as a Service: Challenges, Advances, and Opportunities. *Synthesis Lectures on Information Security, Privacy, and Trust*, 9(3):1–173, Oct 2017.

[151] Yuan Yao, Bin Xiao, Gaofei Wu, Xue Liu, Zhiwen Yu, Kailong Zhang, and Xingshe Zhou. Multi-channel based sybil attack detection in vehicular ad hoc networks using rssi. *IEEE Transactions on Mobile Computing*, 18(2):362–375, 2019.

[152] Chaitanya Yavvari, Zoran Duric, and Duminda Wijesekera. Vehicular dynamics based plausibility checking. In *2017 IEEE 20th International Conference on Intelligent Transportation Systems (ITSC)*, pages 1–8. IEEE, 2017.

[153] Yilu Zhang, G.W. Gantt, M.J. Rychlinski, R.M. Edwards, J.J. Correia, and C.E. Wolf. Connected Vehicle Diagnostics and Prognostics, Concept, and Initial Practice. *IEEE Transactions on Reliability*, 58(2):286–294, Jun 2009.

[154] Junko Yoshida. Toyota Case: Single Bit Flip That Killed. `https://www.eetimes.com/document.asp?doc_id=1319903&page_number=1`. Archived: 2022-05-03, (`https://web.archive.org/web/20220503143858/https://www.eetimes.com/toyota-case-single-bit-flip-that-killed/`).

[155] W. J. Youden. Index for rating diagnostic tests. *Cancer*, 3(1):32–35, Jan 1950.

[156] Ming Yu and Danwei Wang. Model-based health monitoring for a vehicle steering system with multiple faults of unknown types. *IEEE Transactions on Industrial Electronics*, 61(7):3574–3586, 2014.

[157] Mingming Zhang, Chao Chen, Tianyu Wo, Tao Xie, Md Zakirul Alam Bhuiyan, and Xuelian Lin. SafeDrive: Online Driving Anomaly Detection From Large-Scale Vehicle Data. *IEEE Transactions on Industrial Informatics*, 13(4):2087–2096, Aug 2017.

[158] Mingming Zhang, Chao Chen, Tianyu Wo, Tao Xie, Md Zakirul Alam Bhuiyan, and Xuelian Lin. SafeDrive: Online Driving Anomaly Detection From Large-Scale Vehicle Data. *IEEE Transactions on Industrial Informatics*, 13(4):2087–2096, Aug 2017.

[159] Wen Zhang, Yang Wang, Xike Xie, Chuancai Ge, and Hengchang Liu. Real-time Travel Time Estimation with Sparse Reliable Surveillance Information. *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies*, 4(1), Mar 2020.

[160] Hongyang Zhao, Huan Zhou, Canfeng Chen, and Jiming Chen. Join driving: A smart phone-based driving behavior evaluation system. In *GLOBECOM - IEEE Global Telecommunications Conference*, pages 48–53. Institute of Electrical and Electronics Engineers Inc., 2013.