Data Poisoning Attacks in Transportation and Infrastructure-Enabled Defense Methods

Feilong Wang

A dissertation
submitted in partial fulfillment of the
requirements for the degree of

Doctor of Philosophy

University of Washington

2023

Reading Committee:

Xuegang (Jeff) Ban, Chair

Yuan Hong

Edward McCormack

Program Authorized to Offer Degree:

Civil and Environmental Engineering

Abstract

Data Poisoning Attacks in Transportation and Infrastructure-Enabled Defense Methods

Feilong Wang

Chair of the Supervisory Committee:
Xuegang (Jeff) Ban
Department of Civil and Environmental Engineering

Transportation has been and will continue to be under rapid transformations. One transformation is that the transportation system is increasingly driven by massive datasets, including vehicular data (e.g., from GPS, camera and LiDAR) and infrastructure-generated data (e.g., from loop detectors and travel time sensors). However, the growing reliance on data poses potential cybersecurity issues to the transportation system, among which the so-called "data poisoning" attacks by adversaries are becoming increasingly critical. Data poisoning attacks aim to compromise a system's performance by adding malicious noises, perturbations, or deviations to the dataset used by the system. Extensive studies have demonstrated the vulnerability of vehicular data to data poisoning attacks, while research on the infrastructure side is sparse. To narrow the gap, this dissertation develops a generic data poisoning attack model for traffic state estimation and prediction (TSEP) applications that primally rely on infrastructure-generated data. By studying the properties of the proposed attack model such as its Lipschitz conditions, the dissertation identifies the vulnerability of TSEP models, generating insights on proactive defense solutions.

This dissertation then develops defense solutions using secure data from transportation infrastructure, a recent parallel development for supporting various functionalities of advanced vehicle technologies, especially Connected and Automated Vehicles (CAVs). By securing the data collection and transmission processes, secure data from the infrastructure can help defend against

data poisoning attacks. However, such potentials have not been well explored in the literature. This dissertation aims to fill this gap by developing "infrastructure-enabled" solutions to defend against both attacks on vehicular data and attacks on TSEP models. The results show that enabled by the secure infrastructure data, a simple detection method (i.e., an Isolation Forest model) can effectively defend against various types of stealthy GPS spoofing (a high threat to vehicular data of advanced driving systems) and data attacks on queue length estimation models (a common TSEP application).

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF ACRONYMS

AES: Advanced Encryption Standard

AI: Artificial Intelligence

AV: Autonomous Vehicle

BSM: Basic safety message

CAV: Connected and Autonomous Vehicle

CUSUM: Cumulative Summation

CV: Connected Vehicle

C-V2X: Cellular-V2X

DOS: Deny of service

DOT: Department of Transportation

DP: Deep Learning

DRSC: Dedicated Short Range Communication

EKF: Extended Kalman Filter

GPS: Global Positioning System

IED: Infrastructure-Enabled Defense

iForest: Isolation Forest

IMU: Inertial Measurement Unit

iTree: Isolation Tree

ITS: Intelligent Transportation System

KF: Kalman Filter

KKT: Karush–Kuhn–Tucker

LiDAR: Light Detection and Ranging

LS: Least Squared

MAPE: Mean Absolute Percent Error

MNT: MATLAB Navigation Toolbox

MSF: Multi-Sensor Fusion

NESS: Normalized Estimation Error Squared

OBD: On-Board Diagnostics

OCC: One-Class Classifications

OCSVM: One-Class Support Vector Machine

PCA: Principal Component Analysis

RANSAC: Random Sample Consensus

RMSE: Rooted Mean Square Error

RSS: Received Signal Strength

RSU: Roadside unite

SCMS: Secure Credential Management System

SUMO: Simulation of Urban MObility

SVM: Support Vector Machine

TSEP: Traffic State Estimation and Prediction

V2I: Vehicle-to-Infrastructure

V2V: Vehicle-to-Vehicle

V2X: Vehicle-to-everything

VANET: Vehicular ad hoc network

VI: Variation Inequality

VTL: Virtual Time Line

ACKNOWLEDGMENTS

First, I would like to express my sincere gratitude to my advisor and chair of the committee Dr. Xuegang (Jeff) Ban for his invaluable guidance. I would not be able to finish this journey without his support. I would also like to thank all my committee members, including Dr. Yuan Hong (at UConn), Dr. Ed McCormack, and Dr. Bill Howe. They generously devoted their time and provided expertise to help improve this study.

I am also grateful to have many colleagues in iUTS (intelligent Urban Transportation System) Lab and THINK (Transportation-Human Interaction-and-Network Knowledge) Lab for their accompany along this journey. I appreciate their academic advice and life tips. Thanks should also go to many friends and staff at the Department of Civil and Environmental Engineering for their generous help during my study.

I owe tons of thanks to my roommates and many friends. I feel blessed to have these lovely ones sharing their visions, faith, and love during my stay in Seattle.

Last but not least, I express my deep appreciation to my family members. Without their consistent support, I would not be able to join UW and complete my PhD study. Special thanks go to my fiancée, Dr. Rong Zhao, who has been the salt enriching my life and light shining my days over the past six years.

1. INTRODUCTION

1.1. MOTIVATION

Transportation has been and will continue to be under rapid transformation, enabled by recent emerging technologies and systems, such as mobile sensing (Ban and Gruteser, 2012), new mobility systems (Shaheen and Chan, 2016), connected/automated vehicles (CAVs) (Greer et al., 2018), transportation electrification (Shen et al., 2019), and the wide applications of artificial intelligence (AI) in transportation (Cui, 2021). This will profoundly change the landscapes of human mobility, future cities, and our society in general. One of such transformations is the massive old and new datasets that are now widely collected and becoming more available, including vehicular data (e.g., trajectory data from GPS, and sensing data from camera, radar and LiDAR), and infrastructure-generated data (e.g., data from loop detectors and travel time sensors), to name a few. The vehicular data from in-vehicle sensors are essential to support advanced driving systems, while the infrastructure-generated data are primally for *traffic state estimation and prediction (TSEP)* and traffic control, etc.

Besides its benefits, the growing reliance on data poses potential cybersecurity issues to the transportation system by opening various attack surfaces and attack types (Figure 1) (Sheehan et al., 2019). In practice, numerous transportation cybersecurity incidents have been reported, causing safety threats, huge economic losses, and system-level to national-wide disruptions (Callahan, 2019). For example, in 2012, a disgruntled former employee of Texas Auto Center remotely activated over 100 vehicles' immobilization systems and successfully disabled their ignition systems (Linda, 2012). In 2015, hackers announced their success in remotely breaking into a Jeep Cherokee on a highway, demonstrating their control over its steering wheel, accelerator, air-conditioning, radio, windshield wipers, and even brakes (Greenberg, 2015). As a result, 1.4

million vehicles had recalled for installing a security patch. Attackers controlled a fleet of autonomous delivery trucks and rerouted them all to Manhattan at three mph, causing major gridlock in the city (Smith, 2018); an attacker held control of 70% of 187 Washington D.C. traffic and security cameras for ransom (Pegues, 2017). In 2017, FedEx's worldwide operation was hit by NotPetya worm, causing an estimated $300 million in losses (Abt, 2018). UK and German officials reported 2.7 million (in six weeks) attack attempts against their virtual rail transport control system that was purposely set up to gauge better attack frequency and attackers' capabilities (Milne, 2017). Due to these threats, the ITS JPO Strategic Plan 2020–2025 by USDOT recognizes the critical importance of ITS cybersecurity. It encourages state, local, and tribal transportation agencies to identify and mobilize organizational resources to focus on transportation cybersecurity (USDOT, 2020).

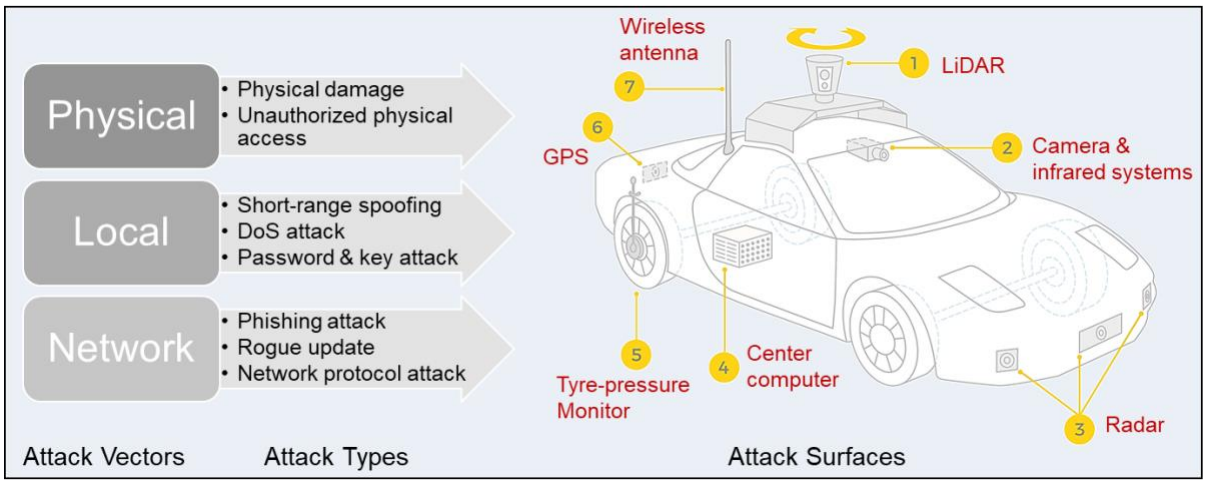

Figure 1. Overview of cyber-attack types, attack vectors and CAV attack surfaces.

Recent years have also witnessed active academic research on transportation cybersecurity, including the vulnerability of transportation systems to various attacks and the development of mitigation solutions. Among these attacks, the so-called "*data poisoning*" attacks are becoming increasingly critical due to their stealthiness. Data poisoning attacks aim to compromise the performance of a system by adding malicious noises, perturbations, or deviations to the dataset

used by the system. For example, GPS spoofing, which falsifies true GPS signals with outdated or maliciously modified GPS signals, has been a long-recognized high threat (Schmidt et al., 2016); LiDAR can be compromised by replay attacks that deceive receivers with recorded and thus outdated data (Cao et al., 2019); vehicular vision systems are vulnerable to attacks that poison the training dataset based on which deep learning-based models (the core of a vision system) are trained (Petit et al., 2015). Note that in cybersecurity and related fields, terms such as invasion (or real-time) attacks (Zhang et al., 2016, 2016) or false data injection attacks (e.g., for the state estimation of electricity grids (Liu et al., 2011)) are also used, while the data poisoning often refers to an attack on the training data. Here, this dissertation overloads the terminology data poisoning as a general term to denote all kinds of data-related attacks in transportation.

Studying attack models is vital for evaluating the vulnerability of data-driven systems and defending against potential attacks from a proactive perspective (Biggio and Roli, 2018). Traditionally, defending attacks have been conducted reactively (Figure 2A): (i) an attacker analyzes a target system and devises an attack strategy to fail the system; and (ii) the defender analyzes the attack and reacts with new countermeasures. It is clear that a reactive approach cannot prevent the never-before-seen types of attacks. By studying attack models against a system, the defender can follow a proactive approach (Figure 2B) by (i) envisioning potential attack scenarios and identifying the vulnerability of the system to attacks, (ii) devising countermeasures against the attacks beforehand, and (iii) repeating this process before system deployment. Therefore, designing attack models is critical for studying transportation cybersecurity.

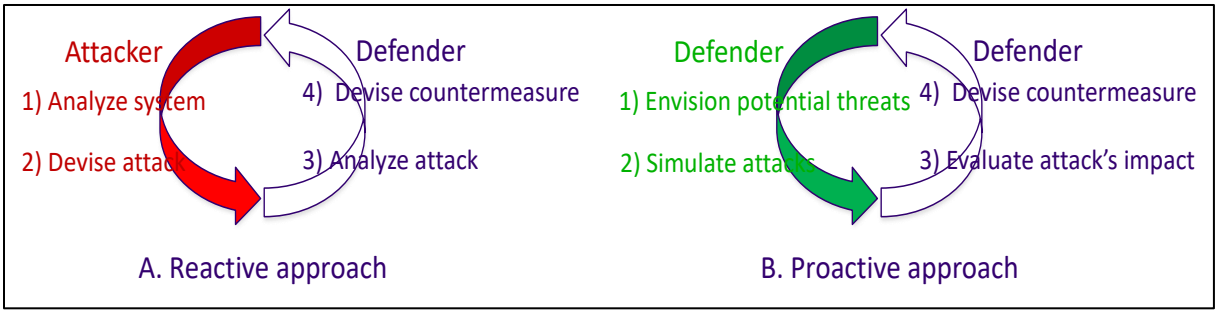

Figure 2. Reactive (A) and proactive (B) approaches to defending against attacks.

In transportation, data poisoning attacks can be broadly categorized as attacks on vehicular data, infrastructure data, and the data transmitted between vehicles and everything (i.e., V2X data). Attacks on vehicular data (for both single and connected vehicles) have been well recognized and extensively studied (Khan et al., 2020; Parkinson et al., 2017; Petit and Shladover, 2015), and there are security mechanisms for the transmitted data (e.g., secure credential management system (SCMS) for V2X data) between vehicles and the infrastructure. On the other hand, research efforts on the infrastructure side are limited. For example, as one of the most important and extensively studied areas in traffic modeling, TSEP uses data at the infrastructure side to produce traffic measures (travel times, volumes, queue lengths, etc.), which are important both in their own merit (e.g., for performance evaluation) and as critical inputs to control and optimization applications (e.g., traffic control). However, data poisoning-related attacks on TSEP have not been formally defined, and few TSEP methods currently consider the risks of data attacks, let alone methods to defend against the attacks.

This dissertation first focuses on the vulnerability of transportation data to data poisoning attacks by formally defining data poisoning attacks against TSEP methods, especially those based on optimization models. Following the analysis of the properties of data poisoning attack models, this dissertation develops defense solutions using emerging transportation infrastructure, a recent parallel development in transportation. Infrastructure is becoming increasingly important in

supporting various functionalities of advanced vehicle technologies, especially CAVs (Y. Li et al., 2020; Wang, 2018). It is widely accepted now (in the transportation field and beyond) that infrastructure-vehicle cooperation is probably a more viable path to implement emerging systems, e.g., automated driving, compared with that using driverless vehicle technologies solely. By securing the data collection and transmission processes, secure data from such infrastructure (different from the one generating data for TSEP) can help defend against data poisoning attacks. Such secure data can be available from existing or newly installed dedicated, secure sensing and data collection systems deployed by the infrastructure (e.g., video cameras installed at an intersection with secure schemes for data transmission and travel times estimation from wired loop detectors). However, such potentials have not been well explored in the literature. This dissertation aims to fill this gap by proposing "infrastructure-enabled" defense solutions (both attack detection and mitigation) detailed below.

1.2. OBJECTIVES

This dissertation aims to propose data poisoning attack models against optimization-based TSEP applications to study their vulnerabilities and develop infrastructure-enabled defense (IED) methods to defend against these attacks. Figure 3 gives an overview of the dissertation's objectives, each specified in the following.

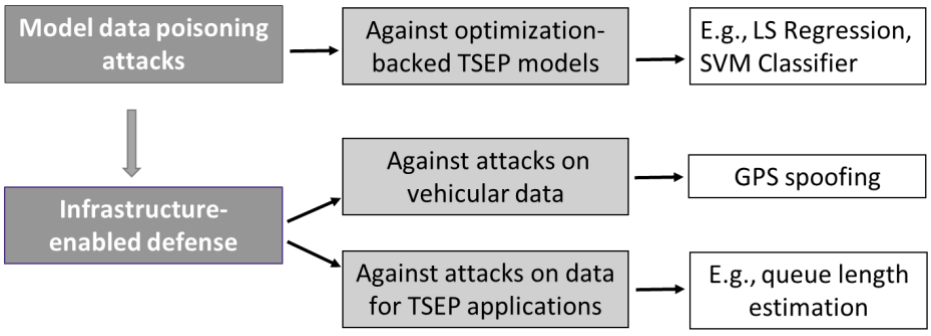

Figure 3. An overview of the objectives of the dissertation

As noted earlier and discussed in more detail in Chapter 2, studies on data poisoning attacks against TSEP models are limited in the literature. The first objective of this dissertation is to narrow this gap by *formally designing data poisoning attacks on TSEP models*. Often formulated as a problem of learning representations from data, many TSEP models are *optimization-based*, including classical linear/nonlinear optimization problems, and most statistical-learning and deep learning-based learning problems. Examples include Least Squares (LS) models applied to delay pattern estimation of signalized intersection using travel times from mobile sensors (Ban et al., 2011), SVM applied to vehicle trajectories for vehicle classification (Sun and Ban, 2013) and freight delivery stop identification (Yang et al., 2014), Bayesian Network-based methods for estimating the cycle-by-cycle queue length distribution (Hao et al., 2014), and DL models for traffic state prediction (W. Li et al., 2020). This dissertation focuses on optimization-based TSEP models. The properties of the proposed attack models will be studied. As shown in Figure 3, the effectiveness will be evaluated using some example TSEP models (e.g., queue length estimation model based on sampled travel time data and vehicle classification model based on trajectory data).

This dissertation then aims to *develop infrastructure-enabled solutions to defend against data poisoning attacks* on transportation data. This is illustrated by developing infrastructure-enabled defense solutions against attacks on *vehicular data* and on *TSEP applications* (Figure 3). For attacks against vehicular data, it takes GPS spoofing as a study case and proposes an infrastructure-enabled defense method, acknowledging that GPS has been and will continue to be an essential sensor in automated driving systems while defending against GPS spoofing has been challenging (Katrakazas et al., 2015). Using GPS spoofing, this dissertation first demonstrates that secure, independent data from the infrastructure enables new ways to detect and mitigate attacks against vehicular data effectively. The infrastructure-enabled solution to defend against attacks on

TSEP models is then studied. The goal is to show how infrastructure status data and existing or newly deployed secure data from the infrastructure can help detect data attacks on TSEP models. This dissertation then investigates the queue length estimation model (an essential model for traffic state estimation and signal control) using mobile travel time data and shows how traffic signal status data and loop detector data could help defend against data attacks on the model.

More specifically, the objectives of this research include:

1. *Study Lipschitz continuity properties of TSEP solution with respect to data perturbations*. This dissertation proposes to formulate data poisoning attacks as the sensitivity analysis of optimization problems with respect to data perturbations (attacks). This intends to open new ways to design data poisoning attacks on various TSEP models. A cross-cutting theme here is to study the Lipschitz continuity properties of the TSEP solution with respect to the data perturbation, which has important implications for designing data attacks and developing defense methods.
2. *Develop a generic attack model and tailor it for common TSEP applications*. Due to the lack of differentiability, the commonly used gradient-based method is not applicable. This study shows how the generalized implicit function theorem (Dontchev and Rockafellar, 2009) can address the challenge due to non-differentiability and general constraints. Effective attack algorithms can then be developed following the computed semi-derivatives. This study also aims to demonstrate *the generality and effectiveness* of the proposed methods via extensive experiments. The proposed methods are tailored for specific TSEP applications. In each specific TSEP model, the performance of the proposed methods is compared with the state-of-the-art methods via extensive experiments.

3. *Develop an infrastructure-enabled solution against GPS spoofing*. The solution utilizes Roadside Units (RSU) as an independent, secure data source. This dissertation considers and addresses the realistic situation where RSU data are not always available for defending against GPS spoofing due to certain constraints (e.g., a limited budget to install RSUs all over the road network). Based on the secure RSU data, an Isolation Forest-based spoofing detection method and a correction model will be developed. The proposed method will be tested using both simulation and real-world data, and its effectiveness is validated in defending against various types of GPS spoofing, including a stealthy attack that is proposed to fail the production-grade autonomous driving systems.
4. *Develop an infrastructure-enabled solution against data poisoning attacks on TSEP models*. After identifying the vulnerability of the TSEP solution to data poisoning attacks, effective defense solutions should be developed. Due to the stealthiness of the data poisoning attacks, conventional anomaly detection methods would be ineffective in distinguishing poisons from normal data, as the two could have the same distribution. This dissertation investigates how infrastructure status data (e.g., traffic signal status data) and secure data from existing or newly deployed infrastructure (e.g., data from wired loop detectors) can help detect data attacks on TSEP. After the attacks are detected, the poisoned data should be removed/tagged to mitigate the (future) impact on the TSEP solution. The performance of the proposed defense solutions shall be evaluated via extensive tests.

1.3. CONTRIBUTIONS

This dissertation brings the cybersecurity of transportation data to the attention of the transportation community. The proposed data poisoning attack models contribute new tools to the community for evaluating the vulnerabilities of data-driven systems/models in transportation. The

infrastructure-enabled defense methods developed in the dissertation provide a promising way to defend against data poisoning attacks, as various (and more intelligent) transportation infrastructure has been developed and continues to emerge as new data sources. Specifically, the major contributions of this dissertation research are listed below.

1) The dissertation fills the research gap by formally formulating and studying data poisoning attacks against TSEP models as a general optimization framework, aiming to identify an optimal perturbation that maximizes the negative impact on the TSEP model. The dissertation shows how the framework can be tailored to maximize the attack's impact on specific TSEP models and datasets. The attacks are designed to be stealthy yet practical by subjecting the attacks to traffic-related constraints (e.g., car-following) and satisfying application-specific knowledge (e.g., vehicles at the rear part of the queue formed during the red time of a traffic signal often experience smaller delays than those in the front part).
2) The properties of the formulated data poisoning attacks are analyzed as sensitivity analysis of optimization problems over data perturbations (attacks), opening a new way to study data poisoning attacks on TSEP models. The innovative way to study the Lipschitz continuity property of the TSEP solution with respect to data perturbations addresses *two major limitations* for most existing attack methods: the ability to deal with constraints, especially inequality constraints and the ability to deal with non-differentiability. Understanding the Lipschitz property of TSEP models has important implications for designing data attacks and developing defense methods.
3) This research designs an infrastructure-enabled solution against GPS spoofing, one of the high threats to vehicular data. The infrastructure-based design has multiple advantages over existing defenses against GPS spoofing. Thus, it provides a new and valuable alternative

to addressing GPS spoofing issues. This research uses secure RSU data and shows that an Isolation Forest-based detection method is more effective than state-of-the-art detection methods. Moreover, it includes a correction component, which corrects location errors induced by the attacks. This is one of the key benefits of the proposed infrastructure-based defense method, as most existing studies only focus on detecting attacks without correcting errors induced by the attacks.

4) This research designs infrastructure-enabled solutions against data poisoning attacks on TSEP models. Given the limited research on defending data poisoning attacks against TSEP models, this research makes a methodological contribution. Focusing on essential TSEP models in practice (e.g., the queue length estimation model), this research also has practical significance to the transportation community. Given its generalizability, the proposed IED solution could also provide useful insights for developing defense solutions to address other data-related transportation cybersecurity issues.
5) This research conducts comprehensive tests and validations of the proposed attack and defense methods in traffic simulation and with real-world data. The data poisoning attack models are tailored and tested in specific TSEP models, including the queue length estimation model represented as a regression problem and the vehicle classification formulated as a binary classification problem. The tests on these TSEP models with distinct properties validate the effectiveness of the proposed data poisoning attack model in practice. The proposed infrastructure-enabled methods are tested under various attack strategies and using both simulation and real-world data to defend against GPS spoofing attacks and attacks on TSEP models. The performance is compared with the state-of-the-art defense strategies, followed by sensitivity analyses.

1.4. DISSERTATION OUTLINE

This dissertation is organized as follows.

Chapter 2 delivers a comprehensive review of the literature on relevant advances and cybersecurity in vehicular technologies and roadside infrastructure, data poisoning attacks, defense methods for data poisoning attacks, and secure infrastructure data.

Chapter 3 gives an overview of the methodological framework of this dissertation.

Chapter 4 presents the work on data poisoning attacks against TSEP models. It starts with the problem statement, introducing the TSEP model as a problem of learning representations from data and formally defining the attacks as a general sensitivity analysis of optimization problems over data perturbations. This chapter studies the Lipschitz continuity property of the TSEP solution under general constraints and lacking differentiability. A generic attack model is proposed following a general implicit function theorem, fitting a broader spectrum of TSEP applications. The proposed methodology is applied and tested to demonstrate its advantages over the state-of-the-art methods. Results are presented and analyzed.

Chapter 5 presents the work on developing infrastructure-enabled solutions against data poisoning attacks in transportation, including two parts. *The first part* focuses on defending against attacks on vehicular data, using GPS spoofing as the case study. The design of secure infrastructure data, GPS spoofing detection and correction methods are introduced. Results from experimental tests using both simulated and real-world GPS data are presented. *The second part* works on defending against attacks on TSEP, using the queue length estimation model investigated in Chapter 4 as an example. The idea of attack detection and correction using signal state data and wired loop detector data is introduced and implemented. By applying the attack model developed in Chapter 4, the performance of the proposed defense solution is evaluated.

Chapter 6 ends the dissertation by discussing the limitations of the present work and future directions.

Chapter 7 concludes the dissertation.

2. LITERATURE REVIEW

2.1. ADVANCES AND CYBERSECURITY IN TRANSPORTATION DATA

2.1.1. Advances in transportation and its reliance on data

Vehicle automation connectivity has been one of the fundamental applications within the field of Intelligent transportation systems (ITS). Enabled by these applications, ITSs increasingly rely on data, including sensor data from automated and connected vehicles and data from roadside infrastructure. The three types of data are introduced in the following.

- Data from automated vehicles

A variety of research projects have advanced the enabling technologies in environmental perception and vehicle control and have produced experimental implementations to show how automation technology could be applied to road vehicles. Automation is achieved using *sensor technology* to survey the environment and to plan vehicle activity (Guerrero-Ibáñez et al., 2018). The state-of-the-art in-vehicle sensors, including camera, radar, LiDAR, etc., have been deployed to support various functions. In particular, Automated Vehicles (AVs) incorporate many different sensors and technologies to enable driverless, safe, and efficient transportation.

Among these advances, accurate *localization* of a vehicle's global positions on the map is critical due to its core role in planning vehicle routes and executing controls (Kuutti et al., 2018). To support high-level driving automation and safety, the localization of AVs needs to be robust in various driving scenarios, including adversarial environments, and thus demands advanced sensors and algorithms. *GPS* has been widely used as a positioning input, combined with other sensors (Shen et al., 2020a; Wang et al., 2021a). The modern localization solutions are predominantly based on data-fusion algorithms that combine position information from multiple sensors,

including GPS, IMU (Inertial Measurement Unit) and LiDAR (Light Detection and Ranging) (Shen et al., 2020a). The localization module fusing various sensors is not only for achieving high accuracy positioning but also for ensuring robustness under different road and weather conditions for safe driving, as each has strengths and weaknesses (Cao et al., 2019; Gao et al., 2015; Shen et al., 2020a). IMUs can generate relative measurements (i.e., accelerations and angular velocities) at high frequency, enabling continuous localization by recursively integrating these measurements. However, since these local measurements contain errors, the location errors will accumulate over time and are not bounded (Min et al., 2019; Noureldin et al., 2009). Therefore, the low-frequent global measurements from GPS or LiDAR are necessary to correct the accumulated errors of IMU. The positioning accuracy of modern GPS receivers can achieve centimeter-level with corrections from ground stations (NovAtel, n.d.; Shen et al., 2020a). However, GPS signals could be affected by surrounding barriers such as tunnels or tall buildings (Hofmann-Wellenhof et al., 2007). LiDAR-based methods localize a vehicle by scanning the surroundings using lasers and matching the scans with High-Definition Map in the database (Gao et al., 2015). However, they are sensitive to weather and changes in surroundings (e.g., buildings or landscapes) (Cao et al., 2019).

- Data from connected vehicles

Another focus of ITS is on the concepts of *connected vehicles* with emerging technologies enhancing connectivity. These concepts are based on the communication of data among vehicles *(V2V)* and/or between vehicles and the infrastructure or any other devices *(V2X)*, which provides the necessary information to implement ITS applications, such as *position, and speed of movement*, etc. (Kutila et al., 2019). The connections and the data transmitted in between support subsequent automation, making it promising to transform the driver's role from actor to monitor or even pure passenger.

The technology of vehicular ad hoc networks (VANET) is one of the most studied topics in ITS. Based on vehicle-to-vehicle and vehicle-to-roadside communications architectures, VANETs can be formed, and information can be relayed among cars to provide various ITS applications. Typical applications include supporting automated braking, disseminating traffic information, enabling vehicle platooning, providing emergency services, etc. In particular, the vehicle platooning is under active research and has shown its promise in stabilizing traffic flows and reducing fuel consumption (Liang et al., 2015).

- Data from road-side infrastructure

Besides the emerging vehicle-based sensors and communication technologies, *Road-side infrastructure* is becoming increasingly important in supporting various functionalities of automated vehicles, especially Connected and Autonomous Vehicles (CAVs) (Y. Li et al., 2020; Wang, 2018). Further, it is widely accepted now in transportation that infrastructure-vehicle cooperation is probably a more viable path to implement automated driving, compared with driverless vehicle technologies solely. For this, the communication and data transmissions between vehicles and infrastructure will play a central role. Indeed, V2X messages (e.g., the basic safety message (BSM)) have already been defined for data transmitted between vehicles and "everything" (including other vehicles, the infrastructure, and other users of the roadway), and secure data transmission schemes (e.g., SCMS) (Brecht et al., 2018)) have also been proposed for V2X data.

Moreover, infrastructure-generated data (e.g., wireless detectors and video-based traffic surveillance) have been playing and will continue to play an increasingly important role in traffic state estimation and predictions (*TSEP*). Infrastructure (e.g., roadside cameras and radar) supplies essential data sources for TSEP to produce traffic measures, including travel times, volumes, delay

pattern estimation, queue lengths, etc. Such measures are crucial for monitoring and evaluating traffic performance and provide critical input to traffic control and optimization applications.

It should be noted that, infrastructure is increasingly applying "crowdsourcing" to collect data from individual users and/or vehicles (e.g., via mobile sensing or connected vehicles), which is also prone to data poisoning attacks. That is, a user or vehicle with either malicious intent or being hacked could send poisoned data (e.g., the trajectory) to the infrastructure, following all communication protocols. Without specially-designed detection and mitigation methods, such poisoned data may be used for training models or making real-time decisions on the infrastructure side, leading to possibly detrimental consequences.

2.1.2. Cybersecurity in transportation data

As discussed above, developing increasingly intelligent transportation inevitably requires various in-vehicle sensors and the vehicle's connection with its neighboring vehicles and infrastructures. However, the increasing use of sensor technologies onboard and connectivity amplifies the exposure of potential vulnerabilities, which can increase the likelihood of future attacks. It is increasingly critical to start evaluating the cybersecurity implications of automated vehicle systems, including those cooperative and noncooperative. Recent studies have revealed plenty of cybersecurity issues that could compromise vehicles' performance.

Studies suggest that *vehicle sensors* are vulnerable to malicious attacks (Petit and Shladover, 2015). A few existing studies have revealed potential vulnerabilities of in-vehicle sensors to malicious attacks (Wang et al., 2021). By overtaking the Controller Area Network (*CAN* bus) or the On-Board Diagnostics (OBD) system, a false injection attack could manipulate speed, acceleration, or angular readings from in-vehicle sensors (Petit and Shladover, 2015). *GPS jamming* can block GPS signals, while *GPS spoofing* injects false information into true GPS

measurements, aiming to mislead vehicles' trajectories (Schmidt et al., 2016). Data poisoning attacks can corrupt neural network models, which *cameras-based modules* rely on for classifying objects on vehicles' paths (Xie et al., 2022a). Replay attacks can prevent *LiDAR* from receiving real-time data, while LiDAR spoofing can inject false obstacles into the point clouds or mask real obstacles (Cao et al., 2019). Among these attacks, GPS spoofing is considered one of the most dangerous attacks and has been receiving various attention, as it is easy to launch, has a high possibility of success, and is difficult to prevent (Khan et al., 2020). More details of GPS spoofing are provided in the next section.

2.2. DATA POISONING ATTACKS

One of the cybersecurity issues is the so-called "*data poisoning*" attacks by adversaries, which are becoming increasingly critical as transportation systems are more and more data-reliant. Data poisoning attacks aim to compromise the performance of a system by adding malicious noises, perturbations, or deviations to the dataset used by the system. Different from conventional data attacks such as DOS or jamming attacks against sensors, emerging data poisoning attacks can be stealthy as they are often carefully crafted and thus can be challenging to defend.

In computer science, data poisoning attacks can be categorized into two groups based on *attack timing*: evasion attacks and poisoning attacks. The former occurs at decision time (e.g., by designing a false data point that can pass the trained model), while the latter is launched at training time to compromise the model learned from the dataset (e.g., by skewing or stretching the anomaly detector so that the future attack appears normal) (Vorobeychik and Kantarcioglu, 2018). Noticing that even the decision time attacks can compromise the model, *this dissertation overloads the term "data poisoning attack"* to facilitate the discussion. That is, this dissertation defines data poisoning attacks as general attacks falsifying the data related to the system's performance.

A *formal definition* of data poisoning attacks typically consists of multiple components, including adversarial objective/motivation, attack vector, attack model and attacker's capability (Reda et al., 2021). The *adversarial objective* defines what the adversary wants to achieve through the attack. The adversary normally tries to find a set of data perturbations that, after being applied to the pristine dataset, would maximize the negative impact on the TSEP models (e.g., the deviation in the TSEP solution) or induce some pre-defined target (e.g., by minimizing the difference between the pristine solution and the target). An *attack vector* is a means (or path) by which the adversary can gain access and manipulate a victim's system either physically or remotely. An *attack model* is to identify a specific strategy for how the attacker manipulates the dataset to reach the adversarial goal via the attack vector. A sophisticated algorithm is often needed to ensure a stealthy (i.e., small perturbations to data) and effective attack. The *Attacker's capability* (or attack budget) specifies the resources an attacker needs to launch a successful attack. Based on the attackers' access and knowledge level of the target model, an attack could be a white-box attack (full access/knowledge) or a black-box attack (no access/knowledge), with a grey-box in the middle. Without suitable resources, carrying an attack becomes difficult.

2.2.1. Model-specific and domain-specific data poisoning attacks

Data poisoning attacks can be model-specific or domain-specific. Attacks that are not designed for a specific model or algorithm, such as random attacks and bandwidth attacks, have limited success rates. In the following, the two categories of attacks are briefly introduced, focusing on data poisoning attacks related to the transportation system.

Various studies have shown the vulnerability of machine learning models to data poisoning attacks. The *model-specific* attacks are optimized for a specific machine learning model or learning algorithm for effective attacks. Such attacks have been proposed and demonstrated on popular

machine learning models, including SVM (Biggio et al., 2013), regression (Jagielski et al., 2018), unsupervised classifiers (Li and Vorobeychik, 2018), and neural networks (Papernot et al., 2016).

Data poison attacks have also been demonstrated in various *domains*. For these attacks, the goal (objective function) and the constraints will be *domain-specific*. For example, in Computer Science, researchers have developed data poison attack models for evaluating the vulnerability of sensing systems and dedicated data-processing or data-learning algorithms. Effective data attacks have been proposed to fail malware and network anomaly detection, spam filtering, computer vision, Portable Document Format (pdf) reader, and recommender systems (Huang et al., 2011; Vorobeychik and Kantarcioglu, 2018), to name a few. In Electricity and Electronics domain, for example, a popular research area is to evaluate the vulnerability of smart power grids facing data poisoning attacks. The smart grid control system relies on an accurate state estimation from sensor measurements, which could be manipulated by attackers (Liu et al., 2011). The data poisoning attacks on transportation TSEP applications studied in this dissertation can also be considered domain-specific attacks.

Data poisoning attacks are also categorized into *model-targeted* and objective-driven attacks, depending on whether the goal is to induce a pre-defined model or maximize certain given objective (e.g., prediction accuracy) (Suya et al., 2021). Most existing poisoning attacks are objective-driven, where the attacker has a specified adversarial objective, such as reducing the forecasting/classification accuracy of the attacked model. Recently, Suya et al. (Suya et al., 2021) proposed a model-targeted attack. Given a target model in mind (i.e., assuming the adversarial knows where to induce the learning model), the model-targeted attack finds a set of poisoning data points iteratively to induce the victim model to that target model as close as possible. Each iteration finds the data point that maximizes the loss difference between the target model and the

intermediate model (i.e., the one learned on the clean data and generated poisoning points from previous iterations). It was proven that the iteration process could eventually induce a model similar to the target model. This attack model does not pay much attention to the constraints (e.g., limits on the TSEP solution to make the attack stealthy) and works under strong assumption by focusing on *convex* models only.

2.2.2. Data poisoning attacks in transportation

The transportation systems are increasingly data-driven. On the vehicle side, advanced driving systems rely on deep-learning networks trained on massive datasets to percept surroundings. On the infrastructure side, smart signal control systems utilize trajectories from connected vehicles to optimize signal timing schemes. A more detailed review is presented in Section 2.1. Consequently, data poisoning attacks can threaten both vehicular and infrastructure data.

- *Data poisoning attack on vehicular data*

Existing studies on data poison attacks mainly focus on vehicular data from in-vehicle sensors. For example, GPS spoofing, which broadcasts falsified GPS signals, has been a long-recognized high threat (Schmidt et al., 2016); LiDAR can be compromised by replay attacks that deceive receivers with recorded and thus outdated data (Cao et al., 2019); cameras are sensitive to blinding attacks that emit light into the camera (Petit et al., 2015; Petit and Shladover, 2015). Besides the decision-time attacks, studies have shown that vehicle sensing devices or systems can also be attacked during training time. Jiang et al., (2020) have shown that the deep learning-based computer vision system deployed on a vehicle can be attacked by feeding adversarial traffic signs into the training dataset. Path routing algorithms can also be poisoned by attacking the crowd-sourced trajectory data (Yu, 2021). A recent study has demonstrated that a data-fusion-based navigation system can be poisoned (Shen et al., 2020b).

Among these threats, GPS is particularly prone to such data attacks as shown by Petit and Shladover (2015), including jamming and spoofing. Considering its importance in supporting modern localization systems, attacks against GPS could lead to severe consequences. GPS jamming can block GPS signals, while GPS spoofing injects false information into true GPS measurements, aiming to mislead vehicles' trajectories (Schmidt et al., 2016). GPS spoofing is a data poisoning attack, while GPS jamming is a typical denial of service (DoS) attack. DoS attacks prevent sensor readings from being received and data poisoning attacks against GPS inject malicious information into the true one. A full taxonomy of various types of attacks can be found in Petit and Shladover (2015), where GPS spoofing) is considered a high threat to vehicles as it is easy to launch, has a high possibility of success, and is difficult to detect (Khan et al., 2020).

Below is a list of common types of GPS spoofing in recent studies (Psiaki and Humphreys, 2016; van Wyk et al., 2020; Wang et al., 2021b):

- *Instant*: One GPS measurement that is unexplainable and significantly different from previous measurements.
- *Noise*: A consecutive sequence of GPS measurements with increased variance. Noise attack occurs across multiple successive sensor readings.
- *Constant bias*: A sequence of GPS measurements with a constant offset from the vehicle's true locations.
- *Gradual drift (stealthy attack)*: A sequence of GPS measurements that are modified to gradually deviate the vehicle from its true trajectory during a period of time.

The references above also discuss the consequences of each type of GPS spoofing attack in detail. The constant bias and gradual drift attacks have received the most attention among these attacks. In particular, the gradual drift attack belongs *to stealthy attacks*, which is more deceptive

than other attacks: it can result in a large deviation between the true and falsified trajectories over time. Recent studies have proposed sophisticated stealthy attacks, making them difficult to detect. For example, a recently stealthy GPS spoofing is proposed to gradually drift the true vehicle position according to its kinematic model (Wang et al., 2021b). Shen et al. (2020) designed a stealthy GPS spoofing attack (named FusionRipper) that could fail production-grade autonomous driving systems (e.g., Baidu's Apollo system) with over 90% success rate. FusionRipper targets the predominantly adopted Multi-Sensor Fusion (MSF) algorithms in practice for localization and injects falsified GPS measurements by exploiting the vulnerability of MSF. Specifically, after profiling MSF vulnerability, FusionRipper performs exponential spoofing, which injects mild deviations at the beginning for gradually compromising the MSF and then aggressive deviations with exponential growths. The deviations injected over time are controlled by two parameters that are tuned according to MSF's configurations.

In scenarios where vehicles are connected via V2V or V2X communication, the effect of data poisoning attacks can go beyond the targeted vehicle and reach its neighboring vehicles. For example, in vehicle platooning scenarios, an attacker could break the platooning stability by falsifying vehicular data and broadcasting the falsified data via the V2V communication (Dadras et al., 2015). Attacks can also be launched to create false jams with ghost vehicles on the roads, affecting the decisions of neighboring vehicles (Yu, 2021). There are also works demonstrating the threat of data poisoning attacks on vehicular ad hoc networks (Boeira et al., 2018).

- *Data poisoning attack on infrastructure data*

Data poisoning-related research on the transportation infrastructure side is sparse. They are limited to specific scenarios and applications. For example, attacks on signal control systems exploit the V2X communication and aim to manipulate V2X-based signal time schemes by poisoning

trajectories of vehicles approaching an intersection (Feng et al., 2018). While pioneering and practically important, such studies do not focus directly on attacks on TSEP (some may mix TSEP attacks with other applications, such as signal control using TSEP as an input). It is fair to say that data poisoning attacks on TSEP have not been formally defined so far, let alone the defense methods. As a result, most existing TSEP models assume benign data (with possible considerations of data errors or even biases) and are thus vulnerable to data poisoning attacks.

Despite the many existing attack models formulated in a bi-level optimization, they have *two major limitations*: the ability to deal with constraints, especially inequality constraints, and the ability to deal with non-differentiability. *Firstly*, there is limited attention on analyzing the attack models in the presence of constraints, which is often true in TSEP applications. This includes physical and domain-knowledge-related constraints, which confine the TSEP solution and its interactions with the data such that the perturbations are carefully bounded to make the attack stealthy and, thus, difficult to detect. The existence of constraints renders it challenging to analyze the response of TSEP solutions to data changes, as an explicit mapping from data to solution could no longer be available in this case. *Secondly*, most existing methods (Biggio et al., 2013) assume that the functions in objective or constraints are differential, based on which gradient methods can be readily applied. This may not be true for TSEP models, whose solutions typically lack differentiability. Consequently, the existing attack methods, especially the commonly used gradient methods, are not applicable. Specifically, the gradient methods i) assume that the solution's response to the data perturbation is differentiable, and ii) require the response to be expressed in a set of *equations* of data perturbation such that the gradient can be estimated. In cases where the assumption of differentiability is weakened, the problem could only be characterized by certain properties such as Lipschitz continuity and sometimes combine that with

the existence of one-sided differentiability (e.g., directional derivatives) (Dontchev and Rockafellar, 2009).

2.3. DEFENSE AGAINST DATA POISON ATTACKS

There are generally two categories of defenses against data poisoning attacks: *detection-based methods* that detect attacks and *proactive methods* that design robust systems to prevent or protect them from attacks.

2.3.1. Detection-based defense

Detecting data attacks and mitigating their adverse effects is one of the mostly applied strategies. Most existing detection methods in the transportation community follow the path of anomaly detection, which detects data attacks by checking abnormal patterns in the received data. For example, to defend against GPS spoofing, traditional detection strategies rely on hardware for monitoring signal power or arrival angle (Shen et al., 2020b). These studies are often formulated as an anomaly detection problem, including model-based (e.g., Kalman Filters) and data-driven approaches (e.g., state-of-the-art unsupervised learning or deep learning methods). Unfortunately, in stealthy data attacks, the patterns of falsified data can be crafted close to those of pristine data, challenging the fundamentals of anomaly detection-based defenses.

In recent years, emerging sensors have promoted studies that detect sensor attacks on vehicles via cross-validating multiple data sources (Huang et al., 2021; Wang et al., 2021a; Wang et al., 2021b). Attack detection is often formulated as an anomaly detection problem. These studies can be categorized into *data-driven* and *model-based* (Yuanzhe Wang et al., 2021a).

The *data-driven* methods rely on previously prepared data to learn a set of patterns or rules, with which the real-time sensor data is determined as benign or adversarial (Huang et al., 2021;

van Wyk et al., 2020). The rules are learned by formulating a supervised learning problem, where a classifier is learned using the labeled training data. The trained classifier serves as the detector to detect whether a sensor is under attack or not (Luo et al., 2021). Such supervised learning algorithms have effectively detected spoofing attacks on real-time localization systems implemented on a wheeled robot (Guerrero-Higueras et al., 2018). Recently, deep learning-based methods have been applied to detecting anomalies in speed sensors (van Wyk et al., 2020). Despite their success in specific applications, these supervised-learning-based detection methods have two limitations (Luo et al., 2021): 1) the training data requires labeled records, which can be challenging to prepare; 2) the trained model may not be generalizable to address new types of attacks that are not represented in the training data. To address these limitations, unsupervised learning methods are proposed to perform one-class classifications (OCC), which do not require labeled data for training (Sanchez-Hernandez et al., 2007). Specifically, a one-class classifier is learned using the unlabeled, primarily benign samples; then, the real-time sensor data is fed into the learned classifier to detect attacks. Wang et al. (2021) propose a One-Class Support Vector Machine (OCSVM) model to detect anomalies in vehicular sensor readings. Though robust in detecting inconsistency among data sources, OCC-based methods do not address another limitation associated with the data-driven methods: the system can detect attacks but may fail to identify the source of attacks. Without identifying which sensor is under attack, it is challenging to design and implement mitigation measures (e.g., isolating the attacked sensor).

Model-based detection methods monitor the difference between the real-time measurement of one sensor with the expected vehicle position derived from the vehicular dynamic model (Abdollahi Biron et al., 2018; Mousavinejad et al., 2020). The basic idea is that if the sensor measurement deviates from the expected value too much, the sensor may be compromised. The

χ^2 test is often used to determine whether the deviation is large enough to claim the sensor being an outlier or under attack (Brumback and Srinath, 1987). The χ^2 test is a statistical test, based on the statistic Normalized Estimation Error Squared (*NEES*) that follows a χ^2 distribution. However, the χ^2 test can be sensitive to sensor noises, resulting in a high rate of false positives (i.e., outliers due to sensor noises are mistaken as attacks). To mitigate this issue, a cumulative sum (CUSUM) discriminator is proposed to detect attacks on GPS and LiDAR (Yuanzhe Wang et al., 2021a). CUSUM detects an attack by inspecting multiple consecutive sensor measurements instead of one measurement only: if the inconsistency between the sensor measurement and the expected vehicle position appears continuously, the sensor is likely under attack. There are multiple limitations with CUSUM in real-world applications. First, it requires two tuning parameters that can be challenging to determine in real-world implementations. Second, being a model-based method, it relies on a predictive model that stealthy attacks could compromise. Specifically, the input of a predictive model can be carefully manipulated so that the generated predictions are corrupted. Then, the features computed from the predictions are no longer reliable indicators of attacks. The numerical experiments later in this dissertation show the weakness of the CUSUM method when facing stealthy attacks. In stealthy data attacks, the patterns of falsified data can be crafted close to those of pristine data, challenging the fundamentals of anomaly detection-based defenses).

Mitigation/correction Focusing on attack detection, existing defenses against data poisoning attacks in transportation have limited discussion on *mitigating* or correcting the errors accumulated by the attack (Abdollahi Biron et al., 2018; Guerrero-Higueras et al., 2018; Pous et al., 2017). The typical strategy is to run a fail-safe mechanism (e.g., handing over control to the human driver) if an attack is detected (Shafaei et al., 2018). However, such a fail-safe mechanism can be costly as it interrupts the system or may not be applicable in certain scenarios (e.g., in a driverless vehicle).

The solution is to deploy multiple sensors, such that an attacked sensor is isolated and the system relies on the rest of the sensors (Gao et al., 2015). For example, a vehicle equipped with GPS and LiDAR will rely on LiDAR for localization if GPS spoofing is detected (Wang et al., 2021b). However, there are some limitations to such solutions. Firstly, as noted above, it can be challenging to identify the attack source (i.e., which sensor is under attack) in the multi-sensor setting, especially when all sensors are vulnerable. Consequently, isolating the attacked sensor is not trivial. Secondly, in the presence of a detection delay, the data fusion framework would have been partially compromised before noticing an attack and isolating the attacked sensor (Shen et al., 2020b). Consequently, the framework needs to be corrected following the sensor isolation, which lacks discussion in the literature. One possible solution is to run a secondary system (e.g., a localization module independent of a GPS sensor) so that the system under attack is isolated and replaced by the secondary system (Shafaei et al., 2018). Yet, deploying and running redundant systems could be economically and computationally costly. Lastly, though rare (but certainly likely) in practice, a coordinated attack targeting multiple sensors would fail these solutions completely.

2.3.2. Proactive defense

Another approach to preventing data poisoning attacks focuses on enhancing data security via, for example, encryption and user authentication. This practice can often be found in scenarios involving V2V or V2X communications. However, this approach may not be practical in specific scenarios. For example, preventing GPS spoofing following this perspective requires significant modifications of the civilian GPS satellite infrastructure (e.g., by enabling cryptographic authentication), which can be challenging and even impractical (Schmidt et al., 2016).

Protection-based defense aims to deter attacks by designing the system in a way that is robust to potential attacks. For example, in Smart Grid Cybersecurity, one can protect the system by identifying a set of measurement devices (e.g., Phasor Measurement Units) and making them immune to the incumbent cyberattacks (e.g. using physical and efficient cryptographic methods) for ensuring the observability of the states (Reda et al., 2021). These protected measurement devices ensure that the state of the grid network is always correctly observed, even if other devices are under attack.

Another research direction is to design a learning algorithm robust to adversarial data samples. Such defense proposals can be classified into two categories: *noise-resilient* learning algorithms and *adversarial-resilient* defenses. The first category has been extensively studied in the field of *robust statistics*, which provides tools to limit the impact of noise and outliers. For example, as an alternative to ordinary linear regression, Huber regression (Huber, 1964) uses a robust loss function that is robust to outliers. RANSAC (Fischler and Bolles, 1981) iteratively samples the data, fits a model on the sub-samples, and identifies outliers as the data points having large residuals to the fitted model. Since principal component analysis (PCA) can be sensitive to outliers, researchers have proposed improved PCA-like methods to identify robust principal components while removing the outliers (Huang et al., 2011). Although robust to noise and outliers, these methods could be compromised under data poisoning attacks where poisoning data are very similar to the true data distribution (i.e., inliers) (Jagielski et al., 2018).

Adversarial-resilient defenses are emerging from the field of adversarial machine learning. They are learning algorithms specifically designed to defend against potential data poisoning attacks. There are generally two types of adversarial-resilient defenses: data sub-sampling-based estimation and trimmed optimization methods (Vorobeychik and Kantarcioglu, 2018). Data sub-

sampling-based estimation learns many models using the same learning algorithm, while each model is based on a random sample of the training dataset. Then the model with the smallest training error is selected. Trimmed optimization-based methods minimize the empirical risk of the learning model while pruning out a small proportion of data points that incur the largest error (Liu et al., 2017). These methods often assume that the adversarial samples have a different distribution from the pristine samples, leading to a distinct estimation of the learning model. When this assumption is invalid in stealthy attacks, the performance of these methods can be impacted. Furthermore, this line of research is evolving in parallel with the development of model-specific attack models (see Section 2.2.1). Recent studies suggest that *preventing* data attacks by design can be challenging, as the progress on defending strategies is often followed by attack models that successfully fail (or compromise) these defenses (Vorobeychik and Kantarcioglu, 2018).

2.4. Methods of Obtaining Secure Infrastructure Data

As noted earlier, secure data from *infrastructure* provide an alternative, promising opportunity for addressing these challenges existing defense solutions encounter. Infrastructure plays an increasingly important role in modern driving systems to facilitate their various advanced functions, such as detecting pedestrians and efficient driving at intersections (Guo et al., 2021; Huang et al., 2021; Wang et al., 2019). The infrastructure can be vulnerable to malicious attacks, including DoS attacks that break communication between vehicles and infrastructure and spoofing attacks that falsify information being communicated (Parkinson et al., 2017). Fortunately, active research has been conducted on securing infrastructure, and practical security strategies are available (Hasan et al., 2020).

Among these strategies, the effective one is to prevent malicious attacks by securing the infrastructure in design. One common prevention strategy is implementing cryptographic schemes

for data transmitted between vehicles and the infrastructure, ensuring message integrity and authenticity. Data collection and transmission can be secured by applying a variety of state-of-the-art secure channels (e.g., HTTPS, SSL, TLS, and FTPS (Alawatugoda, 2015; Nagao et al., 2005; Singh et al., 2017; Zhang et al., 2021)) that apply advanced encryption algorithms (e.g., DES, 3DES, AES, RSA and Blowfish (Bellare and Rogaway, 2005; Jagielski et al., 2018; Nie et al., 2010; Nie and Zhang, 2009)) for data collected from the (existing or new) secure data collection systems. These existing encryption methods can be evaluated in transportation applications and revised, if needed, to fit better transportation scenarios. For example, Brecht et al. (2018) presented a Security Credential Management System (SCMS) for V2X data. SCMS issues digital certificates to vehicles and RSUs to secure communications while maintaining efficient revocation of misbehaving or malfunctioning vehicles. In practice, to promote high security further, the received secure infrastructure data may also be encrypted before storing them (and then decoded before using them), which ensures data security even if the system (hardware) is hacked (Lee et al., 2018). Blockchain is another emerging technique for data security, though investigations are lacking in transportation in terms of their computational efficiency for real-time decisions and flexibility in managing the data in blockchain (Lei et al., 2017; Mollah et al., 2021; Yuan and Wang, 2016).

These existing studies suggest that secure data transmission between vehicles and the infrastructure can be reasonably done. In defending against GPS spoofing, this study applies the state-of-the-art encryption method to set up secure channels for securing and transmitting data between an RSU and its nearby vehicles and test how the process may impact the performance of the spoofing detection and mitigation methods. Specifically, this study implements an Advanced Encryption Standard (AES) scheme (Nielson and Monson, 2019) to encrypt and decrypt the transmitted data with user authentication, which is similar to the SCMS scheme for secure V2X

transmission. The implementation of AES is available in popular, open python packages (Nakov, 2022). Chapter 5.1 gives more details.

3. METHODOLOGY OVERVIEW

This dissertation aims to propose data poisoning attack models against certain TSEP models for evaluating their vulnerability and to develop infrastructure-enabled defense methods for defending the poisoning attacks on transportation data. This chapter provides an overview of the main research methods developed in this research. More detailed discussions of each method are presented in subsequent chapters.

3.1. DATA POISONING ATTACK MODELS AGAINST TSEP

Data poisoning attacks are an emerging research area in the cybersecurity field, with little effort on TSEP models. A TSEP model is often formulated as a data learning problem, which can be optimization-based, statistical-learning-based, or deep learning-based. This dissertation focuses on optimization-based TSEP models. The attacker tries to find a set of perturbations that, after being added to the pristine TSEP data, the estimated/predicted state would be compromised and deviate from the one from the pristine data. The adversarial goal can be specific, for example, to maximize the deviation in the TSEP solution when compared with the pristine solution, as shown in Figure 4. The perturbations should be carefully crafted to be as stealthy as possible and meet constraints in the physical world. Such a set of perturbations could often be identified by solving an optimization problem: the adversarial goal is expressed in the objective function (e.g., maximizing the deviation in the TSEP solution) and the data (perturbation) is the decision variable. Since deriving the TSEP solution from the data itself is an optimization problem, the attack model is a bilevel optimization problem (Biggio et al., 2013; Jagielski et al., 2018).

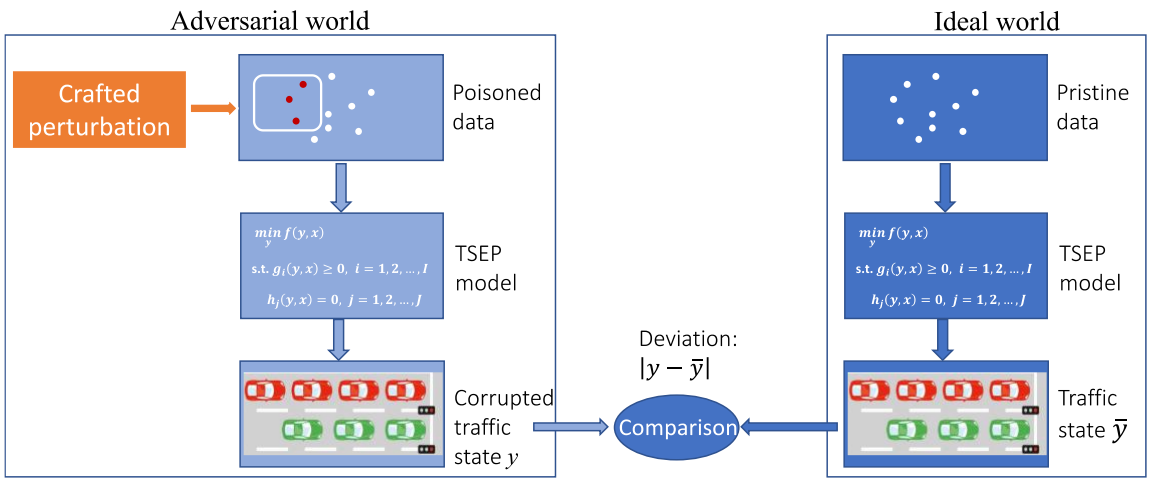

Figure 4. Illustration of data poisoning attack against TSEP model.

Besides the adversary's goal and poisoning attack strategy, a formal definition of the adversarial model also includes a discussion of the adversary's capability (or budget). This component defines whether an attacker has full, partial or no knowledge of the targeted TSEP model and data. It could also specify the attacker's capability in launching the attack. For example, the attack could be limited by upper bounding, such as the number of poisoning samples that can be crafted, or the magnitude of the perturbation added to a sample. The attack may also require carefully bounding the perturbations such that physical constraints and domain knowledge are met, making the attack stealthy and, thus, difficult to detect.

It is crucial to study the sensitivity of the TSEP solution concerning the changes in the data for the purposes of both data poisoning attacks and defenses. Due to the existence of general (equality and inequality) constraints and the lacking differentiability (of the TSEP solution w.r.t. data perturbations), the problem could only be characterized by certain properties such as Lipschitz continuity and sometimes combine that with the existence of one-sided differentiability (e.g., directional derivatives) (Dontchev and Rockafellar, 2009). Therefore, this research focuses on the *Lipschitz continuity* property of the solution w.r.t. the data and develops attack models that fit a

broader spectrum of applications by extending optimization problems with no or equality constraints with problems expressed in variation inequality (VI) with general constraints. The generalized implicit function theorem for VI problems is utilized to compute the semi-derivatives that express how the TSEP solution responds to data perturbations. The semi-derivatives then enable us to evaluate the TSEP models' vulnerability (at each data point) and design a generic attack model. Finally, this study demonstrates the generality and effectiveness of the proposed methodology with applications on common TSEP models. The following chapters will show details of the Lipschitz analysis, derivation of attack models, and solutions to VI problems associated with TSEP models with general constraints, and more numerical results demonstrating the flexibility, generality, and effectiveness of the proposed methodology over state-of-the-art methods.

3.2. INFRASTRUCTURE-ENABLED DEFENSE METHODS

This dissertation discusses potential approaches to defend against data poisoning attacks on transportation data and focuses on developing *infrastructure-enabled defense (IED)* solutions, starting with vehicular data and then infrastructure-generated data for TSEP applications.

For attacks against vehicular data, this dissertation studies GPS spoofing detection and correction. The IED solution to defend against GPS spoofing utilizes *Roadside Units (RSUs)* as an independent, secure data source. The RSUs broadcast secure locational information that vehicles in the broadcast service range can use to estimate their locations periodically. This dissertation considers and addresses the realistic situation where RSU data are not always available due to certain constraints (e.g., a limited budget to install RSUs all over the road network).

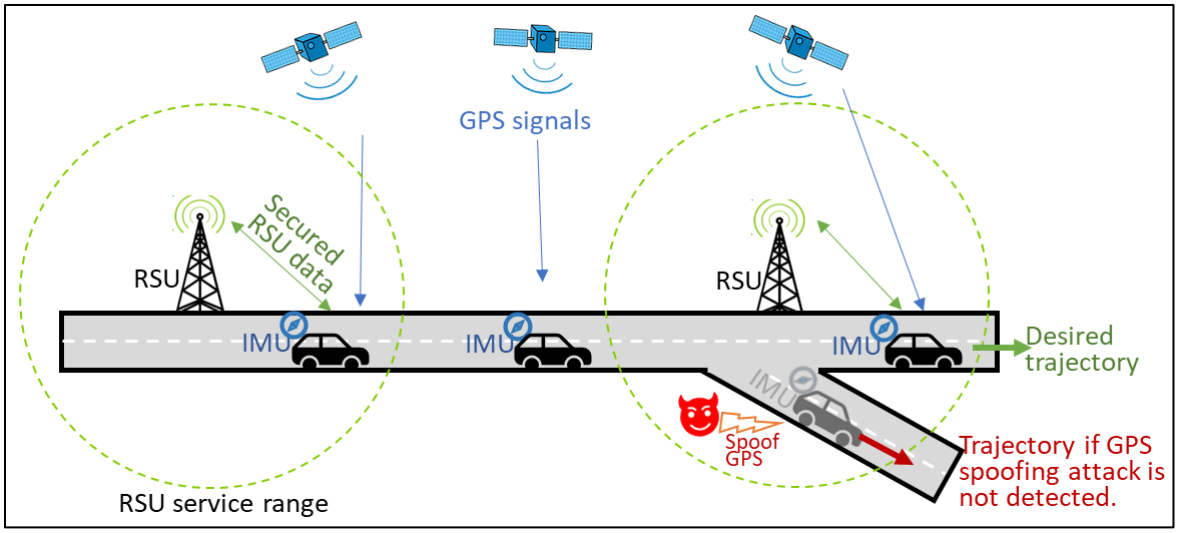

Figure 5. Illustration of the GPS spoofing and the infrastructure-enabled solution.

Figure 5 illustrates the general idea of the infrastructure-enabled solution against GPS spoofing. A vehicle is tracked by a typical motion model with local measurements (e.g., accelerometers and gyroscope) from a low-end IMU as the input and takes global measures from GPS for correcting location errors periodically. GPS measurements can be spoofed. The vehicle could deviate from the desired trajectory without detecting the spoofed GPS measurements. This study develops an infrastructure-enabled solution with which the vehicle can utilize the secure measurements from RSUs to timely detect GPS spoofing and correct location errors resulting from the attacks.

This study first introduces the design of secure RSU data, and the method of how a vehicle interacts with the infrastructure to obtain secure, global position measurements. Based on the secure measurements, multiple features are developed and computed, with which a real-time detector, based on the Isolation Forest, is constructed to detect GPS spoofing. Once spoofing is detected, GPS measurements are isolated, and the potentially compromised location estimator is corrected using the RSU data. This study designs the detection and correction methods under the situation that RSU data are not always available due to certain constraints (e.g., a limited budget

to install RSUs all over the road network). If RSU data are unavailable, an RSU-based prediction model utilizes the last available RSU measurement and the vehicle motion model to predict vehicle locations, preserving timely attack detection. The proposed IED solution is compared with state-of-the-art solutions in defending various types of GPS spoofing, including a stealthy attack that is proposed to fail the production-grade autonomous driving systems (Yuanzhe Wang et al., 2021b).

Similarly, IED solutions to data attacks on TSEP utilize secure data from existing or newly deployed infrastructure. Here the idea is illustrated using the queue length estimation model. The model for estimating queue length at an intersection using sampled travel time data. The data are collected wirelessly and contain the sampled travel times between two consecutive locations: upstream and downstream of a signalized intersection (Figure 6). The preliminary results suggest that the data could be vulnerable to data poisoning attacks, leading to errors in queue length estimation. The proposed defense solution uses infrastructure status data and existing or newly deployed secure data from the infrastructure (e.g., wired loop detectors) to detect attacks on the queue length estimation model (Figure 6).

Similar to defending against GPS spoofing, the proposed solution to attacks against TSEP models includes several components for detection and correction. The first component in the infrastructure-enabled defense method aims to obtain secure estimations of queue lengths from loop detector data and signal status data. The second component runs an attack detector to monitor whether the sampled (mobile) travel time data are poisoned or not. Based on the benchmark data from loop detectors and the signal controller, the attack detector computes multiple features, with which an Isolation Forest model is trained and runs in real-time to detect data poisoning attacks. Once an attack is detected, the potentially compromised queue length is corrected using the one computed from loop detector data. Additionally, the potentially attacked vehicle could be tagged

and reported to the security center that maintains a database to monitor the credibility of vehicles on the road. Each component is described in detail below.

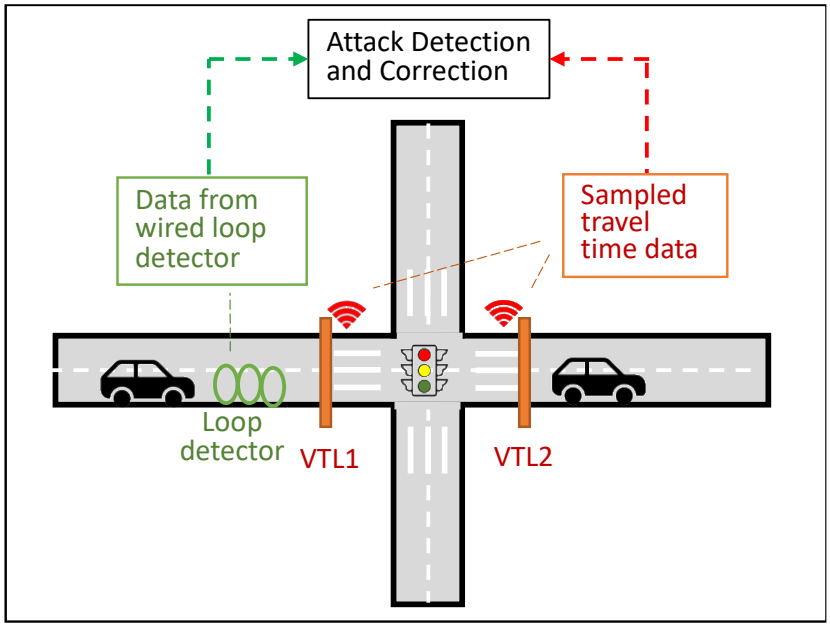

Figure 6. Infrastructure-enabled defense against attacks on queue length estimation model

The proposed defense methods are tested using both simulation data and real-world data. For defending against GPS spoofing, its effectiveness under various types of GPS spoofing is validated, including a stealthy attack that is proposed to fail the production-grade autonomous driving systems. The proposed solution is also compared with the state-of-the-art solutions for GPS spoofing detection and location correction. For defending attacks against TSEP models, the proposed defense is tested on various attack strategies, including perturbing (existing) data points and adding new data points. The performance is evaluated against existing solutions to data poisoning attacks via simulation and real-world data.

4. DATA POISONING ATTACK AGAINST TSEP MODELS

This chapter presents the work on data poisoning attacks against TSEP models. It starts with the problem statement, introducing the TSEP model as a problem of learning representations from TSEP data and some preliminaries. A general framework is then developed to analyze the Lipschitz properties of the TSEP solution with respect to the data. The framework investigates the parameterized version of a nonlinear optimization problem, which allows for studying the behavior of the TSEP solution under the data perturbation (Dontchev and Rockafellar, 2009). Utilizing a generalized implicit function theorem, it develops a poisoning attack model to generate the poisoning points iteratively. Specifically, this generalized implicit function theorem enables us to compute semi-derivatives of the learning parameters with respect to changes in data, which further allows for developing a semi-derivatives-based attack algorithm.

The proposed framework is then tailored to typical TSEP models, including the *LS regression* model (e.g., for queue-length estimation) and *support vector machine (SVM)* for a wide range of transportation applications (e.g., for vehicle classification). As noted earlier, the two models are selected not only because of their broad impact but also their unique properties, allowing us to test the generality of the proposed methods. Though simple, the LS regression model and SVM model are and will continue to be the basis of advanced learning models (e.g., neural networks) and are widely applied in broad applications due to their efficiency, simplicity of use, and effectiveness (Ferrari Dacrema et al., 2019; Tramèr and Boneh, 2021). Therefore, understanding the vulnerability of these fundamental and high-impact models will enable future research on other TSEP models and beyond. Section 5.2 will show that the proposed attack could fail the existing defense solutions (e.g., Huber, RANSAC and TRIM), including those specifically designed for data poisoning attacks.

4.1. PROBLEM STATEMENT, ATTACK MODEL ASSUMPTIONS, AND PRELIMINARIES

4.1.1. Problem Statement

Formal investigation of attack models is important in cybersecurity as it helps reveal the system's vulnerability (or data) and develop effective defense methods. Here the goal is to establish a general framework for TSEP attacks and tailor it to various TSEP applications. Motivated by existing attack models that are often formulated as optimization problems, this work proposes to formulate data poisoning attacks as the sensitivity analysis of optimization problems over data perturbations (attacks). The goal of the attacks is to deviate the TSEP solution by adding perturbations to data used by the TSEP model.

TSEP models take transportation data as input and output the estimated or predicted traffic state. A TSEP model can be optimization-based, statistic-learning-based, or deep-learning-based. This study focuses on the optimization-based TSEP models, which are widely used in transportation. Examples include Least Squares (LS) models applied to delay pattern estimation of signalized intersection using travel times from mobile sensors (Ban et al., 2011), SVM applied to vehicle trajectories for vehicle classification (Sun and Ban, 2013) or freight delivery stop identification (Yang et al., 2014).

Denote y being the learned representation (TSEP solution) from traffic data x. Without loss of generality, this study models a TSEP model under perturbation via a parameterized version of the nonlinear programming problem, which can be written as follows.

$$\begin{aligned} &\hat{y} \in \underset{y}{\operatorname{argmin}}\, g_0(x,y) \\ &s.t. \\ &\quad g_i(x,y) \begin{cases} \leq 0 & \text{for } i \in [1,r] \\ = 0 & \text{for } i \in [r+1,m] \end{cases} \end{aligned} \tag{1}$$

Here, the data x ($\in R^d$) is modeled as the parameter (e.g., data perturbation); as x changes (i.e., poisoned by an adversary), so does the solution of the TSEP model, constructing an attack to the

TESP model. To facilitate the formulation and analysis, this dissertation hereinafter treats this problem (of learning y from x) as a solution mapping P: $\hat{y} \in P(x)$.

Notice that both inequality and equality constraints are included for generality. The TSEP objective $g_0(x, y)$ is often a loss function. The functions expressing the objective and constraints (i.e., $g_0, g_1, \ldots, g_m$) are twice continuously differentiable from $R^d \times R^n$ to R. This study focuses on attacks on models with twice continuously differentiable functions because poisoning attacks in these simple settings are still poorly understood. Moreover, many TSEP models have been relying on models with simple settings as they are easy to implement and maintain and they are often computationally efficient. In specific scenarios, simple models could achieve comparable or better performances than complex deep neural networks (Ferrari Dacrema et al., 2019; Tramèr and Boneh, 2021).

As noted earlier (Figure 4), in an ideal (secure) world, a TSEP model takes the pristine data to estimate/predict traffic state *y*. However, in an adversarial world, some poison samples are crafted, and the input of the TSEP model is corrupted. Consequently, the estimated/predicted state is compromised and deviates from the one from the pristine data. The attack model is to carefully craft such a set of perturbations that the adversarial goal can be achieved while meeting some specific constraints. Finding such a set of perturbations is formulated as solving an optimization problem. Denote $G(x)$ being the adversarial goal, the attack model can be formulated as below:

$$\begin{aligned} &\max_{x} \; G(x) = O(x, \hat{y}) \\ &s.t. \\ &\quad |x - \bar{x}| \leq \delta \\ &\quad \textit{Other application-specific constraints on perturbed data } x \\ &\quad \hat{y} \in P(x) \end{aligned} \tag{2}$$

Here $\bar{x}$ and x are the pristine and poisoned data, respectively. $|x - \bar{x}| \leq \delta$ states that the data perturbations shall be small; the threshold δ would be determined according to the adversary's

capability or the desire to confine the magnitude of perturbations for producing a stealthy attack (i.e., not trigger detection algorithms such as outlier-based anomaly detection methods). Other perturbation-related constraints may vary, e.g., traffic-related constraints or knowledge (e.g., car-following behaviors) should be satisfied.

$G(x)$ can be problem specific. If the goal is to maximize the deviation after the attack, the objective function can be $\max_x G(x) = |\hat{y}(x) - \bar{y}|$, with $\bar{y}$ being the pristine solution (i.e., derived from pristine data). If the goal is to induce the model toward an adversarial target y^*, the objective function is $\min_x G(x) = |\hat{y}(x) - y^*|$. Therefore, the attack model in Eq (2) is more flexible than existing data attacks (Biggio et al., 2013; Jagielski et al., 2018).

4.1.2. Attacks assumptions

A *formal definition* of data poisoning attacks typically consists of multiple components, including adversarial objective/motivation, attack vector, attack model and attacker's capability (Ahmad et al., 2016; Reda et al., 2021).

The *adversarial objective* is the core of defining a data poisoning attack. It varies according to the goal of an adversary. The adversary can try to find a set of perturbations that, after being applied to the pristine dataset, would maximize the negative impact on the TSEP models (e.g., the deviation in the TSEP solution) or induce some pre-defined target.

An *attack vector* is a means (or path) by which the adversary can gain access and manipulate a victim's system either physically or remotely.

Attack model is to identify a specific strategy for how the attacker manipulates the dataset to reach the adversarial goal via the attack vector. A sophisticated algorithm is often needed to ensure a stealthy and effective attack.

Attacker's capability (or attack budget) specifies the resources an attacker needs to launch a successful attack. Based on attackers' access and knowledge level of the target model, an attack could be a white-box attack (full access/knowledge) or black-box attack (no access/knowledge), with the grey-box in the middle. Specifically, in white-box attacks, the attacker is assumed to know the input data, the feature values, and the TSEP model (the learning algorithm or the learned parameters in the model). This setting has been widely considered in previous works for evaluating the vulnerability of machine learning models. In black-box attacks, it is often assumed that the attacker has no or only partial access to the input data but can collect a substitute dataset by exploiting the system. The feature set and TSEP model are known, but the parameters are not (e.g., deep-learning models with hundreds of thousands (and even more) of parameters).

Figure 7 provides a simple example to help understand the attack components. The two data points and the fitted line represents the traffic data and TSEP model, respectively. Assuming the data are collected wirelessly, the attacker compromises the wireless communication channel (i.e., *attack vector*) to poison data $\bar{x}_a$. The goal is to flatten the (negative) slope of the fitted as much as possible (i.e., *adversarial objective*). A white-box attack is assumed so that the attack has full knowledge of the model and data. Nevertheless, the attack cannot change $\bar{x}_a$ wildly but in a limited region (i.e., *attacker's capability*). Given the limit, the attacker identifies an effective algorithm to reach the poison point x'_a that flattens the slope at the best and this step is our attack model.

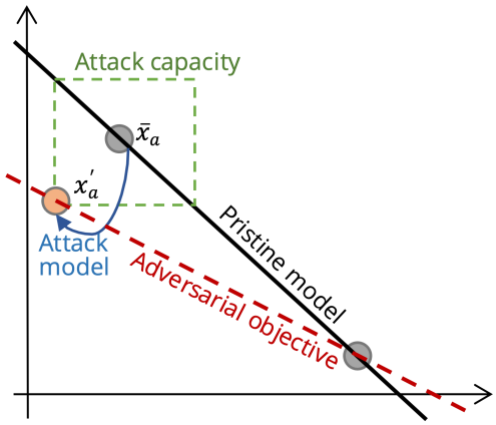

Figure 7. An example illustrating components of a data poisoning attack.

This study focuses on developing and analyzing the attack model given an adversarial objective under some sound (and common) assumptions of the attack vector and attacker's capability. First, this study assumes an adversary can gain full knowledge of training data and model (i.e., white-box attack scenario). Though this assumption may be unrealistic for many scenarios, it allows one to focus on a particular aspect of attacks and is applied in many prior works (Biggio et al., 2013; Koh et al., 2021; Mei and Zhu, 2015; Shafahi et al., 2018). Furthermore, the results from such a setting represent the worst-case outcomes, which could be valuable for evaluating models' vulnerability. Depending on the adversary's capability, his/her capability can be limited by bounding the magnitude of the perturbation and meeting problem-specific constraints to make the attack stealthy. The future study could seek an extension to black-box attacks, depending on the TSEP applications.

Second, the attacker's capability is limited by upper bounding either the number of poisoning samples that can be crafted or the magnitude of the perturbation added to a sample. The attack could also carefully limit the perturbations, such that physical constraints and domain knowledge are met, making the attack stealthy and, thus, difficult to detect.

In terms of the attack vector, this study focuses on attacks *perturbing existing data*, assuming, for example, the communication channel for data transmission (e.g., V2I communication) in TSEP applications is compromised or certain vehicles or devices are attacked from which data are collected. This study later shows that the proposed method can be adapted for the *addition-type attack* to add new poisoning data points (Suya et al., 2021).

4.1.3. Preliminaries

This section introduces relevant concepts and theorems, including the KKT conditions, variational inequality (VI), critical cone, generalized equations, and the general implicit function theorem.

- *KKT conditions of an optimization problem with equity and inequity constraints*

Consider the general optimization problem in Equation (1). Define the Lagrangian function $L(x, \lambda)$. The KKT conditions associated with the problem are listed in the following (Boyd and Vandenberghe, 2004).

- Stationary Condition

$$0 \in \partial_x L(x, \lambda) = \partial_x \left(f(x) + \sum_{i=1}^{m} \lambda_i h_i(x) \right)$$

- Complementary Slackness (only for the r inequality constraints)

$$\lambda_i h_i(x) = 0, \qquad i \in 1, \dots, r$$

- Primal Feasibility

$$g_i(x) \leq 0, \qquad i \in 1, \dots, r$$

$$g_i(x) = 0, \quad i \in r+1, \dots, m$$

- Dual Feasibility (only for the r inequality constraints)

$$\lambda_i \geq 0, \qquad i \in 1, \dots, r$$

- *Definition of VI and Normal Cone*

Some key concepts and their definitions that are helpful in analyzing properties optimization problems are summarized below.

Definition of VI

Given a subset C of R^n *and a mapping* $F: C \to R^n$, $VI(C, F)$ *is to find a vector* y^* *such that*

$$\langle F(y^*),\ y - y^* \rangle \geq 0,\ \forall\, y \in C.$$

The solution set of $VI(C, F)$ is denoted as $SOL(C, F)$.

Definition of normal cone

The *normal cone* of *C* at a point $y^* \in C$ is defined as follows.

$$N_C(y^*) = \{s \mid \langle s,\ y - y^* \rangle \leq 0, \forall\, y \in C\}.$$

Here a "cone" in R^n is a subset of R^n such that it is closed under nonnegative multiplication. That is, set $C \subseteq R^n$ is a cone if $\forall c \in C$, then $\lambda c \in C, \forall \lambda \geq 0$ and λ is a scalar.

From the definitions of VI and normal cone, finding the solution of VI $SOL(C, F)$ is equivalent to finding the solution to the *generalized equation.*

$$0 \in F(y^*) + N_C(y^*),$$

as $-F(y^*) \in N_C(y^*)$ by the definition of VI: $\langle -F(y^*),\ y - y^* \rangle \leq 0,\ \forall\, y \in C$.

The generalized equation reduces to ordinary equations if $C \subseteq R^n$, where $N_C(y^*) = \{0\}$ and the generalized equation holds if and only if $F(y^*) = 0$.

- *Connection between an optimization problem and VI*

There is a close connection between optimization problems and generalized equations. Below is a simple minimization problem.

$$\min_y f(y)$$

$$s.t.\ y \in C.$$

Intuitively, y^* is locally optimal if no descent direction is feasible, where a descent direction $y - y^*$ at y^* means $\langle \nabla f(y^*),\ y - y^* \rangle \leq 0$. Here, assuming f is differentiable, ∇f gives the gradient mapping. Thus, the first-order optimality condition of this problem is given as $\langle \nabla f(y^*),\ y - y^* \rangle \geq 0$. Replacing the mapping F above with ∇f, the first order necessary optimality condition (to furnish y^* a local minimum) can be written in a variational inequality (also termed a *generalized equation* in literature) as follows.

$$0 \in \nabla f(y^*) + N_C(y^*) \text{ or equivalently } -\nabla f(y^*) \in N_C(y^*).$$

A similar formulation can be derived for more general optimization problems (Theorem 2A.7 in (Dontchev and Rockafellar, 2009)). Specifically, a generalized equation expresses the geometric condition that the first-order KKT conditions are trying to express.

This connection states that minimizing a differentiable convex function f over a closed, convex set C is equivalent to solving a certain variational inequality. When $C = R^n$, so that the optimization problem is unconstrainted; this is equivalent to solving $\nabla f(y) = 0$. That is, the variational inequality in a generalized equation reduces to an ordinary equation. Thus, the notion of a variational inequality makes it possible to pass from unconstrained optimization problems to constrained ones.

A similar connection can be made for an optimization problem over sets C that might not be convex and are specified by systems of constraints that can be efficient to handle with Lagrange multipliers. This will lead us to other valuable examples of variational inequalities after some elaborations. Consider the most commonly treated nonlinear programming problem expressed in Equation (1).

The Lagrangian function is defined by

$$L(y,\lambda) = g_0(y) + \langle \lambda, g(y) \rangle = g_0(y) + \sum_{i=1}^{m} \lambda_i g_i(y). \quad (3)$$

If y^* is a local optimizer at which an appropriate condition qualification holds, the KKT necessary condition can be written in a generalized equation following the *Lagrangian Variational Inequalities* theorem below.

Lagrangian Variational Inequalities theorem (Theorem 2A.10 in (Dontchev and Rockafellar, 2009)):

In the minimization problem (Equation (1)), suppose that the set D is a cone, and let Λ be the polar cone D^*,

$$\Lambda = \{\lambda \mid \langle u, \lambda \rangle \leq 0 \ \ for\ all\ \ u \in D\}.$$

Then, in terms of the Lagrangian L in (Equation (3)), the KKT condition necessary condition on y^* and λ can be written in the form below.

$$-\nabla_x L(y,\lambda) \in N_C(y),$$

$$and,\ \nabla_\lambda (y,\lambda) \in N_\Lambda(y),$$

which furthermore can be identified with the variational inequality

$$-F(y,\lambda) \in N_{C\times\Lambda}(y,\lambda)\ ,\ with\ F(y,\lambda) = \Big(\nabla_y L(y,\lambda), -\nabla_\lambda (y,\lambda)\Big).$$

The existence of $y \in C$ satisfying this variational inequality with y is thus necessary for the local optimality of y in problem (Equation (1)) when the constraint qualification is fulfilled. If $L(y,\lambda)$ is convex on C when $\lambda \in \Lambda$, the existence of a λ satisfying this variational inequality with y is moreover sufficient for y to give a global minimum in problem (Equation (1)), without any need of constraint qualification.

4.2. LIPSCHITZ PROPERTY OF TSEP MODELS

For the attack model introduced above, it is crucial to study the sensitivity of the TSEP solution set $P(x)$ with respect to the changes in x. This is the classical (and challenging) sensitivity analysis of optimization problems (Fiacco, 1983). This study focuses on the Lipschitz continuity property of the solution set. Assume x is in a δ-neighborhood of $\bar{x}$, i.e., $|x - \bar{x}| \leq \delta$, and K (the Lipschitz constant) is a finite value. Lipschitz continuity is defined as below.

- If $P(x)$ is single-valued: $P(x)$ is Lipschitz continuous around $\bar{x}$ if $|P(x) - P(\bar{x})| \leq K|x - \bar{x}|$.
- If $P(x)$ is set-valued: $P(x)$ is Lipschitz continuous around $\bar{x}$ if $P(x) \subset P(\bar{x}) + K|x - \bar{x}|B$, with B a unit ball. This study focuses on single-valued mappings.

Note that the above definitions omit some technicalities; one can refer to (Dontchev and Rockafellar, 2009) for more precise definitions. Understanding the Lipschitz property of $P(x)$ is fundamental to both data poisoning attacks and defenses. *For data attacks*, the attack performance will be *bounded* if the Lipschitz property holds, and *unbounded* (potentially explosive) otherwise. An attacker is certainly looking for unbounded attack performance to achieve the largest harm. This, however, mainly depends on the underlying model s/he tries to attack. If there is a choice, the attacker may choose to attack the model for which the Lipschitz property does not hold. *For defenses*, an important implication is that when designing TSEP models, one should be proactive to have the defense in mind and design a TSEP model with the Lipschitz property (so that the attack harm is not unbounded given finite data perturbations), rather than only passively reacting to data attacks. This "*security by design*" concept should be promoted more in the transportation community.

The method for analyzing the Lipschitz properties of TSEP models is introduced in the following.

- *Lagrangian VI and KKT mapping of TSEP models*

Given the parameterized version of nonlinear optimization problems defined in (1), the Lagrangian function is written below.

$$L(x, y, \lambda) = g_0(x, y) + \lambda_1 g_1(x, y) + \cdots + \lambda_m g_m(x, y) \tag{4}$$

Notice that data x is explicitly modeled as a parameter. Then, the VI capturing the associated first-order condition is

$$F(x, y, \lambda) + N_E(y, \lambda) \in (0,0),$$

where,

$$\begin{cases} F(x, y, \lambda) = \left(\nabla_y L(x, y, \lambda), -\nabla_\lambda(x, y, \lambda)\right)^\top, \\ E = R^n \times [R_+^s \times R^{m-s}]. \end{cases} \tag{5}$$

Here, $N_E(y, \lambda)$ computes the normal cone. The pairs (y, λ) satisfying this VI are the KKT pairs for the problem specified by x in Eq (1). If data x is perturbed, one could learn about its influence (mainly the behavior of y) under the perturbation by studying the KKT mapping $S: R^d \rightarrow R^n \times R^m$ that is defined by

$$S(x) = \{(y, \lambda) \mid F(x, y, \lambda) + N_E(y, \lambda) \in (0,0)\}. \tag{6}$$

Note that the optimality condition here generalizes that for a problem with no or only equality constraints. Assuming a problem with m equality constraints, $E = R^n \times R^m$. Then for any $(y, \lambda) \in E$, $N_E(y, \lambda) = \{(0,0)\}$. Therefore, the optimality condition becomes $F(x, y, \lambda) + \{(0,0)\} \in (0,0)$, or simply $\nabla_y L(x, y, \lambda) = 0, -\nabla_\lambda(x, y, \lambda) = 0$. These equations then allow for computing the gradients of (y, λ) over x via the implicit differentiation (or classical implicit

function theorem). Once the gradients are available, the existing gradient-based attack methods can be applied. The same analysis applies to a problem with no constraint.

- *Auxiliary problem*

An auxiliary problem introduced below will be important in studying the behavior of S around a solution pair $(\bar{y}, \bar{\lambda}) \in S(\bar{x})$. Before formulating the auxiliary problem, it is convenient to introduce some notations.

$$\begin{aligned} \bar{g}_0(\Delta y) = &\quad L(\bar{x}, \bar{y}, \bar{\lambda}) + \langle \nabla_y L(\bar{x}, \bar{y}, \bar{\lambda}), \Delta y \rangle + \frac{1}{2} \langle \Delta y, \nabla_{yy}^2 L(\bar{x}, \bar{y}, \bar{\lambda}) \Delta y \rangle \\ \bar{g}_i(\Delta y) = &\quad g_i(\bar{x}, \bar{y}) + \langle \nabla_y g_i(\bar{x}, \bar{y}, \bar{\lambda}), \Delta y \rangle \text{ for } i = 1, \dots, m. \end{aligned}$$

And introduce the notation

$$\begin{aligned} I = &\quad \{i \in [1, m] \mid g_i(\bar{x}, \bar{y}) = 0\} \supset \{s+1, \dots, m\} \\ I_0 = &\quad \{i \in [1, s] \mid g_i(\bar{x}, \bar{y}) = 0 \text{ and } \bar{\lambda}_i = 0\} \subset I \\ I_1 = &\quad \{i \in [1, s] \mid g_i(\bar{x}, \bar{y}) < 0\} \end{aligned}$$

Notice that I gives the set of all the equality constraints and the active inequality constraints; I_0 gives the set of active inequality constraints with Lagrangian multipliers being zero; I_1 gives the set of all the inactive inequality constraints.

The auxiliary problem, depending on a tilt parameter vector v as well as an additional parameter vector $u = (u_1, \dots, u_m)$, is to

$$\begin{aligned} &\min \bar{g}_0(\Delta y) - \langle v, \Delta y \rangle \text{ over all } \Delta y \text{ satisfying} \\ &\bar{g}_i(\Delta y) + u_i \begin{cases} = 0 & \text{for } i \in I \backslash I_0, \\ \leq 0 & \text{for } i \in I_0, \\ \text{unrestricted} & \text{for } i \in I_1. \end{cases} \end{aligned} \tag{7}$$

Notice that now Δy contains the message of how the TSEP solution y responds to the data perturbations.

The Lagrangian for the auxiliary problem depends on the parameter pair (v, u) and involves a multiplier vector $z = (z_1, \dots, z_m)$:

$$L_{aux}(\Delta y, z, v, u) = \bar{g}_0(\Delta y) - \langle v, \Delta y \rangle + \sum_{i=1}^{m} z_i[\bar{g}_i(\Delta y) + u_i].$$

Notice that

$$\begin{array}{rl} \nabla_{\Delta y} L_{aux} = & \nabla_{\Delta y} \bar{g}_0(\Delta y) + \sum_{i=1}^{m} z_i \nabla_{\Delta y} \bar{g}_i(\Delta y) - v \\ -\nabla_{z_i} L_{aux} = & -\bar{g}_i(\Delta y) - u_i \end{array}.$$

The corresponding first-order conditions are given by the VI below.

$$\begin{pmatrix} \nabla_{\Delta y} \bar{g}_0(\Delta y) + \sum_{i=1}^{m} z_i \nabla_{\Delta y} \bar{g}_i(\Delta y) - v \\ -\bar{g}_i(\Delta y) - u_i \text{ for } i \in [1, m] \end{pmatrix} + N_{\bar{E}}(\Delta y, z) \ni \begin{pmatrix} 0 \\ 0 \end{pmatrix},$$

where $\bar{E} = R^n \times W$, with $z = (z_i, \dots, z_m) \ni W \Leftrightarrow z_i \begin{cases} \text{unrestricted} & \text{for } i \in I \backslash I_0, \\ \geq 0 & \text{for } i \in I_0, \\ = 0 & \text{for } i \in I_1. \end{cases}$

This translates into the requirement that

$$\nabla_{yy}^2 L(\bar{x}, \bar{y}, \bar{\lambda}) \Delta y + \sum_{i=1}^{m} z_i \nabla_y g_i(\bar{x}, \bar{y}) - v = 0, \tag{8}$$

$$\text{with } z_i \begin{cases} \geq 0 & \text{for } i \in I_0 \text{ having } \bar{g}_i(\Delta y) + u_i = 0, \\ = 0 & \text{for } i \in I_0 \text{ having } \bar{g}_i(\Delta y) + u_i < 0 \text{ and for } i \in I_1. \end{cases}$$

Note that for $i \in I \backslash I_0$, z_i is unrestricted and the first-order conditions require $z_i[\bar{g}_i(\Delta y) + u_i] = 0$.

Now, the focus is on *the auxiliary solution mapping* $\bar{S}$: $R^n \times R^n \rightrightarrows R^n \times R^m$ that is defined by

$$\bar{S}(v, u) = \{(\Delta y, z) \mid \text{satisfying the requirements in Eq (8)}\},$$

which has

$$(0,0) \in \bar{S}(0,0).$$

- *General implicit function theorem*

The general implicit function theorem for local minima of a parameterized nonlinear programming problem with constraints states as follows (Dontchev and Rockafellar, 2009).

Suppose in the setting of the parameterized nonlinear programming problem (Eq (1)) for twice continuously differentiable functions g_i and its KKT mapping S in Eq (6) that the following conditions hold:

(a) the gradients $\nabla_y g_i(\bar{x}, \bar{y})$ for $i \in I$ are linearly independent.

(b) $\langle \Delta y, \nabla^2_{yy} L(\bar{x}, \bar{y}, \bar{\lambda}) \Delta y \rangle > 0$ for every nonzero $\Delta y \in M^+$, where M^+ is a subspace defined as:

$$M^+ = \{\Delta y \in R^n \mid \Delta y \perp \nabla_y g_i(\bar{x}, \bar{y}) \text{ for all } i \in I \backslash I_0\}$$

Then, firstly, S has a Lipschitz continuous single-valued localization s around $\bar{x}$ for $(\bar{y}, \bar{\lambda})$, and this localization s is semi-differentiable at $\bar{x}$ with semi-derivative given by

$$Ds(\bar{x}; \Delta x) = \bar{s}(-B\Delta x), \text{ where } B = \begin{pmatrix} \nabla^2_{yx} L(\bar{x}, \bar{y}, \bar{\lambda}) \\ -\nabla_x g_1(\bar{x}, \bar{y}) \\ \vdots \\ -\nabla_x g_m(\bar{x}, \bar{y}) \end{pmatrix}.$$

where $\bar{s}$ is a Lipschitz continuous single-valued localization around (0, 0) for (0, 0) in mapping $\bar{S}$.

Secondly, for every x in some neighborhood of $\bar{x}$, the y component of $s(x)$ furnishes a strong local minimum in (Eq (1))). Moreover, conditions (a) and (b) are necessary for the second conclusion when $n + m$ is the rank of the $(n + m) \times d$ matrix B.

Some remarks:

- Condition (a) in the theorem is called the *linear independence constraint qualification condition*, while condition (b) is the *strong second-order sufficient condition*.

- By computing semi-derivative via the auxiliary mapping $\bar{s}$ (from (v, u) to $(\Delta y, z)$), one can evaluate how the TSEP solution (Δy) responds to perturbation Δx. Note that $\bar{s}$'s input (v, u) is specified by $-B\Delta x$.
- Semi-derivative $D_G(\bar{x}; \Delta x)$ depends on the perturbation direction Δx. Then, there exists a direction Δx^* such that $D_G(\bar{x}; \Delta x)$ is maximized, meaning that if one perturbs the data following Δx^* to generate the largest impact on the objective. In practice, Δx^* can be found by identifying all the sub-gradients $d^h_{\Delta x}(\bar{x})$ (such that $D_G(\bar{x}; \Delta x) \ni d^h_{\Delta x}(\bar{x}) \cdot \Delta x$) and picking the one with the largest norm (i.e., $\Delta x^* = \max_h \left| d^h_{\Delta x}(\bar{x}) \right|$). It will be shown later that identifying all the sub-gradients can be straightforward.
- As a special case, one can perturb only one data point at once. Following the previous remark, if point i is the target, there exists Δx_i^* such that $D_G(\bar{x}; \Delta x_i)$ is maximized. Then, there exists a data point a such that $D_G(\bar{x}; \Delta x_a^*) = \max_i D_G(\bar{x}; \Delta x_i^*)$.

4.3. ATTACK STRATEGY

The basic idea of the proposed attack is shown in Algorithm 1 (Figure 8). Sequential perturbations are applied to the pristine dataset: each iteration selects and perturbs one *existing* or *new* data point that has the maximum effect in the direction towards the adversarial (or outer) objective. By repeating this process, the algorithm gradually induces the TSEP solution to reach the adversarial objective (e.g., maximizing the deviation of the TSEP solution or obtaining a target TSEP model specified by the adversarial).

Algorithm 1. Data Poisoning Attack Algorithm

Input: The clean data $x = [x_i]$ ($i \in [1, \dots, n]$) and TSEP model.

Repeat:

1. For $i \in N$, evaluate semi-derivative $D_G(x; \Delta x_i)$, and find

$$D_G^*(x;\Delta x_i) = \max_{\Delta x_i} D_G(x;\Delta x_i), \qquad \Delta x_i^* = \operatorname{argmax}_{\Delta x_i} D_G(x;\Delta x_i)$$

2. Select point a that is the most sensitive to the perturbation

$$a = \operatorname{argmax}_i D_G^*(x;\Delta x_i^*)$$

3. Perturb the target point a following direction Δx_a^* by a specific step size

$$x_a = x_a + \eta \Delta x_a^*.$$

4. Update dataset x and re-learn the TSEP model on the poisoned data

Until: Improvement in the objective value (or the learned parameters) is sufficiently small or constraints are violated.

Output: Poisoned TSEP model

Figure 8. Solution algorithm to the attack model

The key to Algorithm 1 is to compute how the outer objective responds to data perturbations. Denoting composite function $G(x) = O(x, S(x))$, it computes $D_G(x;\Delta x)$, the change of G at x after perturbing it in the direction Δx. The changes in the data impact the solution *explicitly* via the perturbed data x and *implicitly* via shifting the learned representation $y \in S(x)$. This can be expressed following the rule of total differentiation as below.

$$D_G(x;\Delta x) = D_O(x;\Delta x) + D_O\big(S(x); Ds(x;\Delta x)\big) = \nabla_{\Delta x} O + Ds(x;\Delta x)^T \cdot \nabla_S O \qquad (9)$$

In Eq (9), $D_O(x;\Delta x)$ represents the explicit effect of data perturbation on the outer objective at x. On the other hand, $D_O\big(S(x); Ds(x;\Delta x)\big)$ represents the implicit effect of changes in the learned TSEP representation $Ds(x;\Delta x)$ on the objective at $S(x)$. In scenarios where the outer objective does not involve data, the first term is dropped. Given that O is often an explicit, differentiable function of x and $S(x)$, it is relatively easy to find the gradients $\nabla_S O$ and $\nabla_{\Delta x} O$, which represent how the outer objective depends on the data and the learned TSEP representation, respectively.

The challenging part is to find how the data perturbation affects the (TSEP) solution mapping, i.e., $Ds(x;\Delta x)$. It is challenging because, in an optimization problem, an implicit mapping exists from the data to the decision variables (i.e., the learned TSEP parameter in our study). As noted

earlier, previous studies try to address this challenge by finding an implicit differentiation from a set of equations in the KKT conditions. However, in constrained optimization problems, this solution no longer works due to the lacking differentiability. In order to address this challenge, this study follows the Lipschitz analysis and implicit function theorem above and then computes the semi-derivative (in direction Δx) of TSEP solution mapping s at $(\bar{y}, \bar{\lambda})$: $D_s(x; \Delta x) = \bar{s}(-B\Delta x)$.

Also notice that in some applications, there are no poison samples explicitly involved in computing $S(\theta, x)$. This could occur in attacks targeting the model training step, and the trained model is used for estimating or predicting traffic state using non-poisoned data. In such cases, the TSEP solution is impacted only by the corrupted model trained on the poisoned data. This can be expressed as

$$\nabla_{x_a} S(\theta, x) = \nabla_\theta S(\theta, x)|_{x_a} \frac{\partial \theta}{\partial x_a} \tag{10}$$

Once $D_G(x; \Delta x)$ is known, it is straightforward to implement Algorithm 1. In each iteration, the computed semi-derivative allows for evaluating the sensitivity of the outer objective in terms of perturbing each data point so that the most sensitive data point is targeted and perturbed. Specifically, with the computed semi-derivative, the sensitivity at each data point x_i is defined as $\max_{\Delta x_i} D_G(x; \Delta x_i)$. That is, among all the perturbation directions Δx_i applied to data x_i, the largest impact a unit perturbation can have on the outer objective. Once the target is determined, it is perturbed in the optimal direction Δx_i^* (i.e., $\operatorname{argmax}_{\Delta x_i} D_G(x; \Delta x_i)$) by updating the feature vector: $x_a = x_a + \eta \Delta x_a^*$. The perturbation size η (or the step size) could be predefined (when comparing attack effectiveness across different algorithms is desired) or identified via linear search. The algorithm stops when the objective converges (i.e., no sensible change can be observed in sequential iterations).

Although similar iterative attack strategies have been widely adopted in previous studies (Biggio et al., 2013; Jagielski et al., 2018; Mei and Zhu, 2015; Xiao et al., 2015), our attack introduces multiple contributions:

- Most attack models are designed for a specific objective, while our attack is more general in terms of achieving different types of objectives expressed by $O(x, S(x))$.
- Unlike the existing algorithms that assume differentiability and rely on the gradient to find perturbation direction, our attack handles problems with constraints where differentiability is lacking and computes the semi-derivative to identify perturbation direction.
- As shown later, the proposed attack is also flexible as it can be implemented by either perturbing existing data points or adding new poisoning data points.

4.4. TWO TYPICAL TSEP MODELS

The generic framework is tailored to typical TSEP models to maximize its impact, including specifying the objective function in applications, optimization variables, and application-related constraints on *x*. It is also shown how the application-specific knowledge could help design effective attacks (i.e., which data points to attack).

Two popular TSEP models are studied, including the least square (LS) regression model (e.g., for queue-length estimation) and support vector machine (SVM) for a wide range of transportation applications (e.g., for vehicle classification). Besides their high popularity in transportation, the two models are selected for their unique properties and broad impact. The former is regression-based, where both the feature values and the responses of data samples are optimized to launch an attack; it is mainly for real-time traffic state *estimation* with its solution only involving the learned parameters. In contrast, the latter is classification-based, where typically only the feature values are optimized; it is mainly for traffic state *prediction*, with its solution being deviated via the

training process and involving both the learned parameters and the poisoning samples. Though simple, the LS regression model and SVM model are and will continue to be the basis of advanced learning models (e.g., neural networks) and are widely applied in broad applications due to their efficiency, simplicity of use, and effectiveness (Ferrari Dacrema et al., 2019; Tramèr and Boneh, 2021). Therefore, understanding the vulnerability of these fundamental and high-impact models will enable future research on other TSEP models and beyond.

4.4.1. LS regression-based model

- *Lipschitz analysis*

Given the following least squared error problem:

$$\begin{aligned} \min_{y=(\boldsymbol{w},b)} \quad & \frac{1}{2n}\sum_{i=1}^{n}(b+\boldsymbol{w}^{\top}\boldsymbol{x}_i^{ind}-x_i^{dep})^2 \\ \text{s.t.} \quad & \boldsymbol{\beta}y-\boldsymbol{\rho}\leq 0 \end{aligned} \tag{11}$$

where, $\boldsymbol{x}_i=(\boldsymbol{x}_i^{ind},x_i^{dep})$ represents one of the n data points with $\boldsymbol{x}_i^{ind}$ being a vector of length k with independent variables for regression. $\boldsymbol{\beta}$ is a matrix of $m\times(k+1)$ and $\boldsymbol{\rho}$ is a vector of length m, specifying a set of m constraints on y.

The Lagrangian function for Equation (11) is written as

$$L(\boldsymbol{x},y,\boldsymbol{\lambda})=\frac{1}{2n}\sum_{i=1}^{n}(b+\boldsymbol{w}^{\top}\boldsymbol{x}_i^{ind}-x_i^{dep})^2+\sum_{h=1}^{m}\lambda_h(\boldsymbol{\beta}_h^{\top}y-\rho_h). \tag{12}$$

The VI capturing the associated first-order conditions of Equation (11) is given by

$$F(x,y,\lambda)+N_E(y,\lambda)=\begin{pmatrix}\nabla_y L(\boldsymbol{x},y,\boldsymbol{\lambda})\\ -\nabla_{\lambda}L(\boldsymbol{x},y,\boldsymbol{\lambda})\end{pmatrix}+N_E(y,\boldsymbol{\lambda})\ni(0,0). \tag{13}$$

Here,

$$\nabla_y L(x,y,\boldsymbol{\lambda}) = \begin{bmatrix} \nabla_{w_1} L(x,y,\boldsymbol{\lambda}) \\ \vdots \\ \nabla_{w_j} L(x,y,\boldsymbol{\lambda}) \\ \vdots \\ \nabla_{w_k} L(x,y,\boldsymbol{\lambda}) \\ \nabla_b L(x,y,\boldsymbol{\lambda}) \end{bmatrix} = \begin{bmatrix} \frac{1}{n}\sum_{i=1}^{n} \left(b + \boldsymbol{w}^\top \boldsymbol{x}_i^{ind} - x_i^{dep}\right) x_i^{ind_1} + \sum_{h=1}^{m} \lambda_h \beta_{h,1} \\ \vdots \\ \frac{1}{n}\sum_{i=1}^{n} \left(b + \boldsymbol{w}^\top \boldsymbol{x}_i^{ind} - x_i^{dep}\right) x_i^{ind_j} + \sum_{h=1}^{m} \lambda_h \beta_{h,j} \\ \vdots \\ \frac{1}{n}\sum_{i=1}^{n} \left(b + \boldsymbol{w}^\top \boldsymbol{x}_i^{ind} - x_i^{dep}\right) x_i^{ind_k} + \sum_{h=1}^{m} \lambda_h \beta_{h,k} \\ \frac{1}{n}\sum_{i=1}^{n} \left(b + \boldsymbol{w}^\top \boldsymbol{x}_i^{ind} - x_i^{dep}\right) + \sum_{h=1}^{m} \lambda_h \beta_{h,k+1} \end{bmatrix} \tag{14}$$

$$\nabla_\lambda L(x,y,\boldsymbol{\lambda}) = \boldsymbol{\beta} y - \boldsymbol{\rho}$$

$$E = \mathbb{R}^{k+1} \times \mathbb{R}_+^m.$$

The solution pairs $(y, \boldsymbol{\lambda})$ satisfying this VI are the KKT pairs for the problem specified by the perturbation in data x in (3). The influence of changes in $\boldsymbol{x}$ on the solution can be then studied using the KKT mapping $S: \mathbb{R}^d \to \mathbb{R}^d \times \mathbb{R}_+^m$ defined by the VI that captures the associated first-order conditions.

$$S(\boldsymbol{x}) = \{(y, \boldsymbol{\lambda}) | F(\boldsymbol{x}, y, \boldsymbol{\lambda}) + N_E(y, \boldsymbol{\lambda}) \ni (\mathbf{0}, \mathbf{0})\}. \tag{15}$$

Auxiliary problem

An auxiliary problem defined around the solution pair $(\bar{y}, \bar{\lambda}) \in S(\bar{\boldsymbol{x}})$ is written as below.

$$\begin{aligned} \min_{\Delta y = (\Delta w, \Delta b)} \quad & L(\bar{\boldsymbol{x}}, \bar{y}, \bar{\boldsymbol{\lambda}}) + \langle \nabla_y L(\bar{\boldsymbol{x}}, \bar{y}, \bar{\lambda}), \Delta y \rangle + \frac{1}{2} \langle \Delta y, \nabla_{yy}^2 L(\bar{\boldsymbol{x}}, \bar{y}, \bar{\lambda}) \Delta y \rangle - \langle v, \Delta y \rangle \\ & = L(\bar{\boldsymbol{x}}, \bar{y}, \bar{\lambda}) + \frac{1}{2} \langle \Delta y, \begin{bmatrix} \boldsymbol{\Sigma} & \mathbf{0} \\ \mathbf{0} & 1 \end{bmatrix} \Delta y \rangle - \langle v, \Delta y \rangle \\ \text{s.t.} \quad & \boldsymbol{\beta}_h^\top \bar{y} - \rho_h + \boldsymbol{\beta}_h^\top \Delta y + u_h \begin{cases} = 0 \text{ if } \boldsymbol{\beta}_h^\top \bar{y} - \rho_h = 0 \text{ and } \bar{\lambda}_h > 0 \\ \leq 0 \text{ if } \boldsymbol{\beta}_h^\top \bar{y} - \rho_h = 0 \text{ and } \bar{\lambda}_h = 0 \\ \text{Free} \quad \text{if } \boldsymbol{\beta}_h^\top \bar{y} - \rho_h < 0 \end{cases} \\ & h = [1, \ldots, m] \end{aligned} \tag{16}$$

Here, $\boldsymbol{\Sigma} = \frac{1}{n}\sum_{i=1}^{n} \boldsymbol{x}_i^{ind} \boldsymbol{x}_i^{ind\top} = \frac{1}{n}\sum_{i=1}^{n} \left(\begin{bmatrix} x_i^{ind_1} \\ \vdots \\ x_i^{ind_k} \end{bmatrix} \times \begin{bmatrix} x_i^{ind_1} & \ldots & x_i^{ind_k} \end{bmatrix} \right)$ is a covariance matrix if the data is feature-wise normalized (i.e., $\mu_{j(j\in[1,\ldots,k])} = \frac{1}{n}\sum_{i=1}^{n} x_i^{ind_j} = 0$). And the Hessian matrix $\nabla_{yy}^2 L(\bar{x}, \bar{y}, \bar{\lambda})$:

$$\nabla^2_{yy}L(\bar{x},\bar{y},\bar{\lambda}) = \begin{bmatrix} \frac{1}{n}\sum_{i=1}^{n} x_i^{ind_1}x_i^{ind_1} & \cdots & \frac{1}{n}\sum_{i=1}^{n} x_i^{ind_1}x_i^{ind_j} & \cdots & \frac{1}{n}\sum_{i=1}^{n} x_i^{ind_1}x_i^{ind_k} & \frac{1}{n}\sum_{i=1}^{n} x_i^{ind_1} \\ \vdots & \ddots & \vdots & \ddots & \vdots & \vdots \\ \frac{1}{n}\sum_{i=1}^{n} x_i^{ind_j}x_i^{ind_1} & \cdots & \frac{1}{n}\sum_{i=1}^{n} x_i^{ind_j}x_i^{ind_j} & \cdots & \frac{1}{n}\sum_{i=1}^{n} x_i^{ind_j}x_i^{ind_k} & \frac{1}{n}\sum_{i=1}^{n} x_i^{t_j} \\ \vdots & \ddots & \vdots & \ddots & \vdots & \vdots \\ \frac{1}{n}\sum_{i=1}^{n} x_i^{ind_k}x_i^{ind_1} & \cdots & \frac{1}{n}\sum_{i=1}^{n} x_i^{ind_k}x_i^{ind_j} & \cdots & \frac{1}{n}\sum_{i=1}^{n} x_i^{ind_k}x_i^{ind_k} & \frac{1}{n}\sum_{i=1}^{n} x_i^{t_k} \\ \frac{1}{n}\sum_{i=1}^{n} x_i^{t_1} & \cdots & \frac{1}{n}\sum_{i=1}^{n} x_i^{t_j} & \cdots & \frac{1}{n}\sum_{i=1}^{n} x_i^{t_k} & 1 \end{bmatrix}$$
$$= \begin{bmatrix} \boldsymbol{\Sigma} & \boldsymbol{\mu} \\ \boldsymbol{\mu}^T & 1 \end{bmatrix}$$

The Lagrangian function for the auxiliary problem is written as:

$$\begin{aligned} &L_{Aux}(\Delta y, z, v, u) \\ &= \bar{L}(\Delta y, z) - \langle v, \Delta y\rangle + \langle z, u\rangle \\ &= L(\bar{x}_a, \bar{y}, \bar{\lambda}) + \frac{1}{2}\langle \Delta y, \begin{bmatrix} \boldsymbol{\Sigma} & \boldsymbol{\mu} \\ \boldsymbol{\mu}^T & 1 \end{bmatrix}\Delta y\rangle + \sum_{h=1}^{m} z_h(\boldsymbol{\beta}_h^\top \bar{y} - \rho_h + \boldsymbol{\beta}_h^\top \Delta y + u_h) - \langle v, \Delta y\rangle + \langle z, u\rangle \end{aligned} \tag{17}$$

Here $\boldsymbol{z} = (z_1, \ldots, z_{\mathrm{m}})$ stands for Lagrangian variables.

The corresponding first-order conditions for the auxiliary problem are given by the VI below

$$\begin{pmatrix} \nabla_{\Delta y} L_{Aux}(\Delta y, z, v, u) \\ -\nabla_z L_{Aux}(\Delta y, z, v, u) \end{pmatrix} + N_{\bar{E}}(\Delta y, z) \ni (0,0)$$

Here, $L_{Aux}(\Delta y, z, v, u)$ is Lagrangian function for the auxiliary problem with

$$\nabla_{\Delta y} L_{Aux}(\Delta y, z, v, u) = \begin{bmatrix} \boldsymbol{\Sigma} & \boldsymbol{0} \\ \boldsymbol{0} & 1 \end{bmatrix}\Delta y + \sum_{h=1}^{m} z_h \boldsymbol{\beta}_h - \boldsymbol{v},$$

$$\nabla_z L_{Aux}(\Delta y, z, v, u) = \boldsymbol{\beta}^\top \bar{y} - \boldsymbol{\rho} + \boldsymbol{\beta}^\top \Delta y + \boldsymbol{u},$$

and $\bar{E} = \mathbb{R}^{k+1} \times W$, with

$$\boldsymbol{z} = (z_1, \ldots, z_m) \in W \Leftrightarrow z_h \begin{cases} \geq 0 \ \text{ if } \boldsymbol{\beta}_h^\top \bar{y} - \rho_h = 0 \text{ with } \bar{\lambda}_h = 0 \\ = 0 \qquad\qquad \text{if } \boldsymbol{\beta}_h^\top \bar{y} - \rho_h < 0 \\ Free \ \text{ if } \boldsymbol{\beta}_h^\top \bar{y} - \rho_h = 0 \text{ with } \bar{\lambda}_h > 0 \end{cases}$$

The VI above can be translated into the requirements below.

$$\begin{bmatrix} \boldsymbol{\Sigma} & \boldsymbol{0} \\ \boldsymbol{0} & 1 \end{bmatrix}\Delta y + \sum_{h=1}^{m} z_h \boldsymbol{\beta}_h - \boldsymbol{v} = 0,$$

$$\text{with} \quad z_h \begin{cases} \geq 0 & \text{if } \boldsymbol{\beta}_h^\top \bar{y} - \rho_h = 0 \text{ with } \bar{\lambda}_h = 0 \text{ having } \boldsymbol{\beta}_h^\top \Delta y + u_h = 0 \\ = 0 \ \text{ if } \boldsymbol{\beta}_h^\top \bar{y} - \rho_h = 0 \text{ with } \bar{\lambda}_h = 0 \text{ having } \boldsymbol{\beta}_h^\top \Delta y + u_h < 0 \text{ OR } \boldsymbol{\beta}_h^\top \bar{y} - \rho_h < 0 \\ Free & \text{if } \boldsymbol{\beta}_h^\top \bar{y} - \rho_h = 0 \text{ with } \bar{\lambda}_h > 0, \text{then from KKT}, z_h\big(\boldsymbol{\beta}_h^\top \Delta y + u_h\big) = 0 \end{cases} \tag{18}$$

Then the auxiliary solution mapping $\bar{S}\colon \mathbb{R}^d \times \mathbb{R}^m \rightrightarrows \mathbb{R}^d \times \mathbb{R}^m$ is defined by

$$\bar{S}(v,u) = \{(\Delta y, z) \mid \text{satisfying Eq (18)}\} \tag{19}$$

Sufficient conditions for Lipschitz continuity

Following the general implicit function theorem, the (two) sufficient conditions are checked to ensure that *S has a Lipschitz continuous single-valued s*.

a) The gradients $\nabla_y g_h(x,y)$ for all active constraints are linearly independent;

Since $\nabla_y g_h(\boldsymbol{x},y) = \boldsymbol{\beta}_h$ in our case, this is to check whether vectors in

$$\{\boldsymbol{\beta}_h | \boldsymbol{\beta}_h^\top \bar{y} - \rho_h = 0, h \in [1,\ldots,m]\}$$

are linearly independent. This condition thus depends on the specific problem formulation and the solution $\bar{y}$.

b) $\frac{1}{2}\langle \Delta y, \nabla_{yy}^2 L(\bar{\boldsymbol{x}},\bar{y},\bar{\lambda})\Delta y\rangle > 0$ *for every nonzero* $\Delta y \in M^+$.

It can be shown that

$$\nabla_{yy}^2 L(\bar{\boldsymbol{x}},\bar{y},\bar{\lambda}) = \begin{bmatrix} \boldsymbol{\Sigma} & \mathbf{0} \\ \mathbf{0} & 1 \end{bmatrix}$$

is positive definitive, and this condition is naturally satisfied. As noted earlier,

$$\boldsymbol{\Sigma} = \frac{1}{n}\sum_{i=1}^{n} \boldsymbol{x}_i^t \boldsymbol{x}_i^{t\top} = \frac{1}{n}\sum_{i=1}^{n} \left(\begin{bmatrix} x_i^{t_1} \\ \vdots \\ x_i^{t_d} \end{bmatrix} \times \begin{bmatrix} x_i^{t_1} & \ldots & x_i^{t_d} \end{bmatrix} \right)$$

is a covariance matrix and $\boldsymbol{\mu} = \frac{1}{n}\sum_{i=1}^{n} \boldsymbol{x}_i^t = \mathbf{0}$. (Note as a common data pre-processing step, the multi-dimensional data are feature-wise normalized to have $\mu_j = \frac{1}{n}\sum_{i=1}^{n} x_i^{t_j} = 0$ $(j = 1,\ldots,k)$).

According to the property of Schur complement, a matrix in the form of $\boldsymbol{X} = \begin{bmatrix} A & B \\ B^T & C \end{bmatrix}$ possesses a nice property:

If C is invertible, then $\boldsymbol{X}$ is positive definite if and only if C and its Schur complement $X/C = A - BC^{-1}B^T$ are both positive definite.

Apply this property to our case, and it can be observed that $C = 1$ (invertible and positive definite) and $X/C = \boldsymbol{\Sigma} - \mathbf{0} \times 1 \times \mathbf{0}^T = \boldsymbol{\Sigma}$. Therefore, $\nabla^2_{yy} L(\bar{x}, \bar{y}, \bar{\lambda})$ shall be positive definite if $\boldsymbol{\Sigma}$ is positive definite.

Being a covariance matrix, $\boldsymbol{\Sigma}$ is surely positive semi-definite; furthermore, it shall be positive definite as long as there exists no feature in the data that is a linear combination of other features (i.e., $\boldsymbol{x}^{t_{j'}} = \sum_{j \neq j'} \boldsymbol{x}^{t_j}$), which is uncommon in a data modeling process. Therefore, it safe to conclude that $\nabla^2_{yy} L(\bar{x}, \bar{y}, \bar{\lambda})$ shall be positive definite in cases that $\boldsymbol{w} = (w_1, \ldots, w_k)$.

Therefore, according to the general implicit function theorem, *the mapping S has a Lipschitz continuous single-valued localization s around $\bar{x}_a$ for* $(\bar{y}, \bar{\lambda})$ and this localization s is semidifferentiable at $\bar{x}_a$.

- *Attack algorithm*

As noted earlier, one needs to compute $D_G(\overline{\boldsymbol{x}}; \Delta x) = \nabla_{\Delta x} O + Ds(\overline{\boldsymbol{x}}; \Delta x)^T \cdot \nabla_S O$ under the perturbation Δx. $\nabla_{\Delta x} O$ and $\nabla_S O$ can be easily derived for a specific objective function O; the key is to find out $Ds(\overline{\boldsymbol{x}}; \Delta x)$ (i.e., how the learned parameters respond to the data perturbations). Assuming data point a is perturbed, the $Ds(\overline{\boldsymbol{x}}; \Delta x_a)$ is computed as $\bar{s}(-B\Delta x_a)$ following the implicit function theorem. Specifically,

$$B = \begin{pmatrix} \nabla^2_{yx_a} L(\bar{x}_a, \bar{y}, \bar{\lambda}) \\ -\nabla_{x_a} g_i(\bar{x}_a, \bar{y}) (i = 1, \ldots, m) \end{pmatrix} = \begin{bmatrix} \boldsymbol{\kappa}_1 = \frac{1}{n} \left[\overline{\boldsymbol{x}}_a^{ind} \overline{\boldsymbol{w}}^\top + (\bar{b} + \overline{\boldsymbol{w}}^\top \overline{\boldsymbol{x}}_a^{ind} - \bar{x}_a^{dep}) \boldsymbol{I}_k \quad -\overline{\boldsymbol{x}}_a^{ind} \right] \\ \boldsymbol{\kappa}_2 = \frac{1}{n} [\overline{\boldsymbol{w}}^\top \quad -1] \\ \mathbf{0}_{m \times d} \end{bmatrix}$$

Note that $-B\Delta x_a = -[\boldsymbol{\kappa}_1, \boldsymbol{\kappa}_2, \mathbf{0}]^T q$ specifies mapping $\bar{s}$'s input (v, u). By inserting the input into Eq (18) and with some rearrangement, Δy can be obtained by solving the following equation.

$$\begin{aligned}
\Delta y \quad &= \begin{bmatrix} \Delta w \\ \Delta b \end{bmatrix} \\
&= \nabla^2_{yy} L(\bar{\boldsymbol{x}}, \bar{y}, \bar{\boldsymbol{\lambda}})^{-1} [v - \sum_{h=1}^{m} z_h \boldsymbol{\beta}_h] \\
&= -\begin{bmatrix} \boldsymbol{\Sigma} & \mathbf{0} \\ \mathbf{0} & 1 \end{bmatrix}^{-1} (\begin{bmatrix} \boldsymbol{\kappa}_1 \\ \boldsymbol{\kappa}_2 \end{bmatrix} \Delta x_a + \sum_{h=1}^{m} z_h \boldsymbol{\beta}_h)
\end{aligned} \tag{20}$$

$$\text{with} \quad z_h \begin{cases} \geq 0 & \text{if } \boldsymbol{\beta}_h^\top \bar{y} - \rho_h = 0 \text{ with } \bar{\lambda}_h = 0 \text{ having } \boldsymbol{\beta}_h^\top \Delta y = 0 \\ = 0 & \text{if } \boldsymbol{\beta}_h^\top \bar{y} - \rho_h = 0 \text{ with } \bar{\lambda}_h = 0 \text{ having } \boldsymbol{\beta}_h^\top \Delta y < 0 \text{ OR } \boldsymbol{\beta}_h^\top \bar{y} - \rho_h < 0 \\ Free & \text{if } \boldsymbol{\beta}_h^\top \bar{y} - \rho_h = 0 \text{ with } \bar{\lambda}_h > 0, \text{then from KKT,} \quad z_h \boldsymbol{\beta}_h^\top \Delta y = (\end{cases}$$

4.4.2. Support Vector Machine

- *Lipschitz analysis*

The TSEM problem is:

$$\begin{aligned}
\min_{y=(\boldsymbol{w},b,\boldsymbol{\xi})} \quad & \frac{1}{2} \| \boldsymbol{w} \|_2^2 + C \sum_{i}^{n} (\xi_i)^o \\
\text{s.t.} \quad & g_i^\alpha = 1 - \xi_i - l_i(\boldsymbol{x}_i^\top \boldsymbol{w} + b) \leq 0 \\
& g_i^\mu = -\xi_i \leq 0 \\
& i = 1, \ldots, n
\end{aligned} \tag{21}$$

Here, $\boldsymbol{x}_i = [x_{i,1}, \ldots, x_{i,k}]$ gives k feature values of one of the n data points and l_i gives the label (1 or -1 as one of two classes of vehicles). C is a hyper-parameter chosen by the user; a larger C means to assign a higher penalty to errors. The objective function is a convex programming problem for any positive integer o; previous studies often take $o = 1$ or 2 (Burges, 1998; Diehl and Cauwenberghs, 2003). This study works on $o = 1$ which enhances the robustness to outliers with the hinge loss and thus is generally preferred in practice. The choice $o = 1$ also has the advantage that neither the ξ_i nor their Lagrange multipliers appear in the dual problem, making it convenient to analyze properties of the SVM model.

The Lagrangian function can be written as

$$L(x, y, \boldsymbol{\alpha}, \boldsymbol{\mu}) = \frac{1}{2} \parallel \boldsymbol{w} \parallel_2^2 + C \sum_i^n \xi_i + \sum_i^n (\alpha_i [1 - \xi_i - l_i(\boldsymbol{x}_i^\top \boldsymbol{w} + b)]) - \sum_i^n \mu_i \xi_i, \qquad (22)$$

where α_i, μ_i $(i \in [1, n])$ are the multipliers.

From the solution structure of the SVM model, the data points can be grouped into multiple groups. Denoting N being the full group containing all the points, this study introduces five subgroups of points to facilitate our discussion.

$$\begin{aligned}
&\alpha_i = 0 \Leftrightarrow \mu_i = C \Rightarrow g_i^\mu = 0 \Rightarrow \begin{cases} g_i^\alpha = 0 & R_0 \\ g_i^\alpha < 0 & R_1 \end{cases} \\
&0 < \alpha_i < C \Leftrightarrow 0 < \mu_i < C \Leftrightarrow g_i^\mu = 0 \Rightarrow g_i^\alpha = 0 \quad V \\
&\alpha_i = C \Leftrightarrow \mu_i = 0 \Rightarrow g_i^\alpha = 0 \Rightarrow \begin{cases} g_i^\mu = 0 & E_0 \\ g_i^\mu < 0 & E_1 \end{cases}
\end{aligned} \qquad (23)$$

Notice that g_i^α is inactive for points $i \in R_1$ while active for the rest of the points. In particular, g_i^α defined at $i \in R_0$ are active with the multipliers $\alpha_i = 0$. g_i^μ is inactive for points $i \in E_1$ while active for the rest. In particular, g_i^μ defined at $i \in E_0$ are active with the multipliers $\mu_i = 0$. These observations will be used for deriving the auxiliary problem below.

The VI that captures the associated first-order conditions can be written as

$$F(x, y, \alpha, \mu) + N_E(y, \alpha, \mu) = \begin{pmatrix} \nabla_y L(x, y, \alpha, \mu) \\ -\nabla_\alpha L(x, y, \alpha, \mu) \\ -\nabla_\mu L(x, y, \alpha, \mu) \end{pmatrix} + N_E(y, \alpha, \mu) \ni (\mathbf{0}, \mathbf{0}, \mathbf{0})$$

or more specifically

$$\begin{pmatrix} w_j - \sum_i^n (\alpha_i l_i x_{i,j}) \quad (j = [1, \ldots, k]) \\ -\sum_i^n (\alpha_i l_i) \\ C - \alpha_i - \mu_i \quad (i \in N) \\ -1 + \xi_i + l_i(x_i^\top w + b) \quad (i \in N) \\ \xi_i \quad (i \in N) \end{pmatrix} + N_E(y, \alpha, \mu) \ni (0,0,0). \qquad (24)$$

Here,

$$E = \mathbb{R}^{k+1+n} \times (\mathbb{R}_+^n \times \mathbb{R}_+^n)$$

The influence of changes in $\boldsymbol{x}$ on the solution can be studied using the KKT mapping S: $\mathbb{R}^k \rightarrow \mathbb{R}^2 \times \mathbb{R}^{k+1+n} \times \mathbb{R}^n \times \mathbb{R}^n$ defined by

$$S(x_a) = \{(y, \alpha, \mu) | F(x, y, \alpha, \mu) + N_E(y, \alpha, \mu) \ni (\mathbf{0}, \mathbf{0}, \mathbf{0})\}. \tag{25}$$

Then an auxiliary problem can be defined around the solution pair $(\bar{y}, \bar{\alpha}, \bar{\mu}) \in S(\bar{x}_a)$. First, notice that

$$\nabla_{yy}^2 L(x, y, \alpha, \mu) \quad = \nabla_y \begin{bmatrix} w_j - \sum_i^n (\alpha_i l_i x_{i,j}) \quad (j = [1, \ldots, k]) \\ -\sum_i^n (\alpha_i l_i) \\ C - \alpha_i - \mu_i \quad (i \in N) \end{bmatrix} = \begin{bmatrix} \mathbb{I}_k & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} \end{bmatrix}$$

with, $\mathbb{I}_k$ stands for the identity matrix of $d \times d$.

$$\begin{aligned} \frac{1}{2}\langle \Delta y, \nabla_{yy}^2 L(\bar{x}_a, \bar{y}, \bar{\alpha}, \bar{\mu}) \Delta y \rangle &= \frac{1}{2} \Delta w^\top \Delta w \\ \langle \nabla_y g_i^\alpha(\bar{x}_i, \bar{y}), \Delta y \rangle &= -l_i x_i^\top \Delta w - l_i \Delta b - \Delta \xi_i \cdot \\ \langle \nabla_y g_i^\mu(\bar{x}_i, \bar{y}), \Delta y \rangle &= -\Delta \xi_i \end{aligned} \tag{26}$$

Here, $\Delta y = (\Delta w, \Delta b, \Delta \xi_{i \in N})$.

Let

$$\begin{aligned} \bar{g}_0(\Delta y) &= L(\bar{x}, \bar{y}, \bar{\alpha}, \bar{\mu}) + \langle \nabla_y L(\bar{x}, \bar{y}, \bar{\alpha}, \bar{\mu}), \Delta y \rangle + \frac{1}{2}\langle \Delta y, \nabla_{yy}^2 L(\bar{x}, \bar{y}, \bar{\alpha}, \bar{\mu}) \Delta y \rangle \\ &= L(\bar{x}, \bar{y}, \bar{\alpha}, \bar{\mu}) + \frac{1}{2} \Delta w^\top \Delta w \\ \bar{g}_i^\alpha(\Delta y) &= g_i^\alpha(\bar{x}_i, \bar{y}) + \langle \nabla_y g_i^\alpha(\bar{x}_i, \bar{y}), \Delta y \rangle \\ &= g_i^\alpha(\bar{x}_i, \bar{y}) + (-l_i x_i^\top \Delta w - l_i \Delta b - \Delta \xi_i) \\ \bar{g}_i^\mu(\Delta y) &= g_i^\mu(\bar{x}_i, \bar{y}) + \langle \nabla_y g_i^\mu(\bar{x}_i, \bar{y}), \Delta y \rangle \\ &= g_i^\mu(\bar{x}_i, \bar{y}) + (-\Delta \xi_i) \end{aligned}$$

The auxiliary problem is to

$$\min_{\Delta y = (\Delta w, \Delta b, \Delta \xi_{i(i \in N))})} \bar{g}_0(\Delta y) - \langle v, \Delta y \rangle \text{ over all } \Delta y \text{ satisfying} \tag{27}$$

$$\bar{g}_i^{\alpha}(\Delta y) + u_i^{\alpha} \begin{cases} = 0 & \text{if } i \in V \cup E_0 \cup E_1 \\ \leq 0 & \text{if } i \in R_0 \\ free & \text{if } i \in R_1 \end{cases}$$

$$\bar{g}_i^{\mu}(\Delta y) + u_i^{\mu} \begin{cases} = 0 & \text{if } i \in V \cup R_0 \cup R_1 \\ \leq 0 & \text{if } i \in E_0 \\ free & \text{if } i \in E_1 \end{cases}$$

$$i = [1, \dots, n]$$

Notice that now Δy (denoted w in the book) contains the message of how the TSEP solution responds to the perturbations.

Lagrangian for the auxiliary problem can be written as:

$$\begin{aligned} & L_{Aux}(\Delta y, z_i^{\alpha}, z_i^{\mu}, v, u) \\ & = \bar{L}(\Delta y, z_i^{\alpha}, z_i^{\mu}) - \langle v, \Delta y \rangle + \langle z, u \rangle \\ & = L(\bar{x}_a, \bar{y}, \bar{\alpha}, \bar{\mu}) + \frac{1}{2}\Delta w^{\top}\Delta w + \sum_i^n z_i^{\alpha}(\bar{g}_i^{\alpha}(\bar{x}_a, \bar{y}) + (-l_i x_i^{\top}\Delta w - l_i \Delta b - \Delta\xi_i)) \\ & + \sum_i^n z_i^{\mu}(\bar{g}_i^{\mu}(\bar{x}_a, \bar{y}) + (-\Delta\xi_i)) - \langle v, \Delta y \rangle + \langle z, u \rangle \end{aligned} \tag{28}$$

Here, Lagrangian variables $z_{i\in N}^{\alpha}$ correspond to constraints $g_{i\in N}^{\alpha}$ and $z_{i\in N}^{\mu}$ correspond to constraints $g_{i\in N}^{\mu}$.

The corresponding first-order conditions for the auxiliary problem are given by the variational inequality

$$\begin{aligned} & \begin{pmatrix} \nabla_{\Delta y} L_{Aux}(\Delta y, z, v, u) \\ -\nabla_{z_i^{\alpha}} L_{Aux}(\Delta y, z, v, u)(i \in N) \\ -\nabla_{z_i^{\mu}} L_{Aux}(\Delta y, z, v, u)(i \in N) \end{pmatrix} + N_{\bar{E}}(\Delta y, z) \ni (0,0,0) \\ & \text{with, } \nabla_{\Delta y} L_{Aux}(\Delta y, z, v, u) = \begin{bmatrix} \Delta w_j - \sum_i^n z_i^{\alpha} l_i x_{i,j} \quad (j = [1, \dots, k]) \\ -\sum_i^n z_i^{\alpha} l_i \\ -z_i^{\alpha} - z_i^{\mu} \quad (i \in N) \end{bmatrix} - v \\ & -\nabla_{z_i^{\alpha}} L_{Aux}(\Delta y, z, v, u) = -\bar{g}_i^{\alpha}(\bar{x}_i, \bar{y}) + l_i(x_i^{\top}\Delta w + \Delta b) + \Delta\xi_i - u_i^{\alpha} \\ & -\nabla_{z_i^{\mu}} L_{Aux}(\Delta y, z, v, u) = -\bar{g}_i^{\mu}(\bar{x}_i, \bar{y}) + \Delta\xi_i - u_i^{\mu} \end{aligned} \tag{29}$$

The VI for the auxiliary problem can be translated into the specific requirements that

$$\begin{bmatrix} \Delta w_j - \sum_i^n z_i^\alpha l_i x_{i,j} \quad (j = [1, \ldots, k]) \\ -\sum_i^n z_i^\alpha l_i \\ -z_i^\alpha - z_i^\mu \quad (i \in N) \end{bmatrix} - v = 0, \text{ with}$$

$$z_i^\alpha \begin{cases} \geq 0 & \text{for } i \in R_0 \text{ having } -l_i(x_i^\top \Delta w + \Delta b) - \Delta\xi_i + u_i^\alpha = 0 \\ = 0 & \text{for } i \in R_0 \text{ having } -l_i(x_i^\top \Delta w + \Delta b) - \Delta\xi_i + u_i^\alpha < 0 \text{ and } i \in R_1 \\ free & \text{for } i \in V \cup E_0 \cup E_1, \end{cases} \tag{30}$$

$$z_i^\mu \begin{cases} \geq 0 & \text{for } i \in E_0 \text{ having } -\Delta\xi_i + u_i^\mu = 0 \\ = 0 & \text{for } i \in E_0 \text{ having } -\Delta\xi_i + u_i^\mu < 0 \text{ and } i \in E_1 \\ free & \text{for } i \in V \cup R_0 \cup R_1 \end{cases}$$

$$i = [1, \ldots, N]$$

Note that the computation here follows the facts that $\bar{g}_i^\alpha(\bar{x}_a, \bar{y}) = 0$ if $i \in R_0$ and $\bar{g}_i^\mu(\bar{x}_a, \bar{y}) = 0$ if $i \in E_0$.

The auxiliary solution mapping $\bar{S}$: $\mathbb{R}^{k+1+n} \times \mathbb{R}^{2n} \rightrightarrows \mathbb{R}^{k+1+n} \times \mathbb{R}^{2n}$ is then defined by

$$\bar{S}(v, u) = \{(\Delta y, z) \mid \text{satisfying Eq (30)}\}$$

Sufficient conditions for Lipschitz continuity

Following the Implicit Function Theorem introduced earlier, two sufficient conditions are checked to ensure that *S has a Lipschitz continuous single-valued s*.

a) The gradients $\nabla_y g_i(\bar{x}_a, \bar{y})$ for all active constraints are linearly independent.

The gradients consist of

$$\begin{array}{lccccccccl} & w & & b & \xi_1 & \ldots & \xi_i & \ldots & \xi_n & \\ \nabla_y g_i^\alpha(\bar{\boldsymbol{x}}, \bar{y}) = & [-l_i \boldsymbol{x}_i & & -l_i & 0 & \ldots & -1 & \ldots & 0] & \text{for} i \in R_0 \cup V \cup E_0 \cup E_1 \\ \nabla_y g_i^\mu(\bar{\boldsymbol{x}}, \bar{y}) = & [\ 0 & & 0 & 0 & \ldots & -1 & \ldots & 0] & \text{for} i \in E_0 \cup V \cup R_0 \cup R_1 \end{array} \tag{31}$$

This condition depends on the data. In practice, these vectors are unlikely linearly dependent, based on several observations from real-world modelings as below.

- R_0 and E_0 stand for rare cases (where constraints are active and Lagrangian multipliers are zero) and are often empty in real-world datasets.
- A vector in $\nabla_y g_i^{\alpha}(\bar{x}_a, \bar{y})$ for $i \in E_1$ and $\nabla_y g_i^{\mu}(\bar{x}_a, \bar{y})$ for $i \in R_1$ is independent with other vectors (i.e., cannot be expressed as a linear combination of other vectors). Our attention turns to checking whether there is a $j \in V$ having the following linear combination:

  $\nabla_y g_j^{\alpha}(\bar{x}_a, \bar{y}) = \sum_{i \in V \cup E_1, j \neq i} \beta_i^{\alpha} \times \nabla_y g_i^{\alpha}(\bar{x}_a, \bar{y}) + \sum_{i \in V \cup R_1} \beta_i^{\mu} \times \nabla_y g_i^{\mu}(\bar{x}_a, \bar{y})$.

  Our focus is to verify whether there is a vector $[-l_j x_j - l_j]$ $(j \in V)$ can be expressed as a linear combination of other vectors in V (i.e., $[-l_i x_i - l_i]$ $(i \in V, i \neq j)$), as it is unlikely to use any vectors $[-l_i x_i - l_i]$ $(i \in E_1 \cup R_1)$ to express $[-l_j x_j - l_j]$. Otherwise, the vectors from $E_1 \cup R_1$ introduce nonzero elements to the right side of $\nabla_y g_j^{\alpha}(\bar{\boldsymbol{x}}, \bar{y})$ (in Equation (31)) and there is no way to cancel them out.

  There is a low chance of expressing $[-l_j x_j - l_j]$ using $[-l_i x_i - l_i]$ belonging to V, especially when V contains only a few points (the feature of SVM).

In applications, the linearly independent is checked via numerical methods. The test in the next section shows that the vectors are indeed linearly independent of the vehicle classification dataset.

b) $\frac{1}{2}\langle \Delta y, \nabla_{yy}^2 L(\bar{x}_a, \bar{y}, \bar{\alpha}, \bar{\mu}) \Delta y \rangle > 0$ for every nonzero $\Delta y \in M^+$.

Noticing that

$$\frac{1}{2}\langle \Delta y, \nabla_{yy}^2 L(\bar{x}_a, \bar{y}, \bar{\alpha}, \bar{\mu}) \Delta y \rangle = \frac{1}{2} \Delta w^{\top} \Delta w \tag{32}$$

The following analysis shows that if Δy is nonzero, Δw shall be nonzero, which will yield $\frac{1}{2}\Delta w^{\top}\Delta w > 0$.

First, in subspace M^{+}, it requires $\Delta y = (\Delta w, \Delta b, \Delta\xi_{i(i\in N)})$ to be perpendicular to every gradient as follows:

$$\begin{array}{lccccccccll} & w & & b & \xi_1 & \dots & \xi_i & \dots & \xi_n & & \\ \nabla_y g_i^{\alpha}(\bar{\boldsymbol{x}}_a, \bar{y}) = & [-l_i \boldsymbol{x}_i & & -l_i & 0 & \dots & -1 & \dots & 0] & \text{for } i \in V \cup E_0 \cup E_1 & (1) \\ \nabla_y g_i^{\mu}(\bar{\boldsymbol{x}}_a, \bar{y}) = & [\ 0 & & 0 & 0 & \dots & -1 & \dots & 0] & \text{for } i \in V \cup R_0 \cup R_1 & (2) \end{array} \tag{33}$$

With (37-2), it can be derived that $\Delta\xi_i = 0$ for $i \in V \cup R_0 \cup R_1$.

If $\Delta w = 0$, from (37-1) $(\Delta b, \Delta\xi_{i(i\in E_0\cup E_1)})$ satisfies that

$$\Delta y^{\top}\nabla_y g_i^{\alpha} = -l_i\Delta b = 0, \text{for every } i \in V \text{ (notice } \xi_{i\in V} = 0)$$

and

$$\Delta y^{\top}\nabla_y g_i^{\alpha} = -l_i\Delta b - \Delta\xi_i = 0, \text{for every } i \in E_0 \cup E_1,$$

which means $\Delta b = 0$ and $\Delta\xi_i = 0$ for $i \in E_0 \cup E_1$, i.e., $\Delta y = 0$.

This concludes that Δw cannot be zero if Δy is required to be nonzero. Therefore, condition (b) is indeed satisfied.

It is easy to find that these two conditions shall be satisfied as well when $o = 2$ (i.e., the slack variable ξ_i are added to the objective of SVM in the quadratic term), as the only difference lies in the computation of $\frac{1}{2}\langle\Delta y, \nabla_{yy}^2 L(\bar{x}_a, \bar{y}, \bar{\alpha}, \bar{\mu})\Delta y\rangle$ that is given by

$$\frac{1}{2}\langle\Delta y, \nabla_{yy}^2 L(\bar{x}_a, \bar{y}, \bar{\alpha}, \bar{\mu})\Delta y\rangle = \frac{1}{2}\Delta w^{\top}\Delta w + \frac{1}{2}\Delta\boldsymbol{\xi}^{\top}\Delta\boldsymbol{\xi},$$

which is again certainly positive for every nonzero $\Delta y \in M^{+}$.

Therefore, according to the general implicit function theorem, *the mapping S has a Lipschitz continuous single-valued localization s around* $\bar{x}_a$ *for* $(\bar{y}, \bar{\alpha}, \bar{\mu})$ and this localization s is semidifferentiable at $\bar{x}_a$.

- *Attack algorithm*

Again, to compute $D_G(\bar{\boldsymbol{x}}; \Delta x) = \nabla_{\Delta x} O + Ds(\bar{\boldsymbol{x}}; \Delta x)^T \cdot \nabla_S O$ under the perturbation Δx, the key is to find out $Ds(\bar{\boldsymbol{x}}; \Delta x)$. Assuming data point a is perturbed, $Ds(\bar{\boldsymbol{x}}; \Delta x_a)$ can be computed as $\bar{s}(-B\Delta x_a)$. Specifically,

$$B = \begin{pmatrix} \nabla^2_{yx_a} L(\bar{x}_a, \bar{y}, \bar{\alpha}, \bar{\mu}) \\ -\nabla_{x_a} g_i^{\alpha}(\bar{x}_a, \bar{y}) \text{ for } i \in N \\ -\nabla_{x_a} g_i^{\mu}(\bar{x}_a, \bar{y}) \text{ for } i \in N \end{pmatrix} \tag{34}$$

with,

$$\begin{aligned} \nabla^2_{yx_a} L(\bar{x}_a, \bar{y}, \bar{\alpha}, \bar{\mu}) &= [-\bar{\alpha}_a l_a \mathbb{I}_k \quad 0 \quad 0]^T \\ \nabla_{x_a} g_i^{\alpha}(\bar{x}_a, \bar{y}) &= \begin{cases} -l_a \bar{w} & \text{if } i = a, \\ 0 & \text{otherwise,} \end{cases} \\ \nabla_{x_a} g_i^{\mu}(\bar{x}_a, \bar{y}) &= 0 \qquad \text{for } i \in N \end{aligned} \tag{35}$$

Noticing that $-B\Delta x_a$ specifies $\bar{s}$'s input (v, u) and many elements in (v, u) are zero, the input to Eq (30) leads the following

$$\begin{aligned} \Delta w &= \bar{\alpha}_a l_a \Delta x_a + \textstyle\sum_i^n z_i^{\alpha} l_i x_i \\ \textstyle\sum_i^n z_i^{\alpha} l_i &= 0 \qquad\qquad\qquad\qquad \text{, with} \\ z_i^{\alpha} &= -z_i^{\mu} \quad \text{for } (i \in N) \end{aligned}$$

$$z_i^{\alpha} \begin{cases} \geq 0 & \text{for } i \in \mathrm{R}_0 \text{ having } -l_i(x_i^{\top} \Delta w + \Delta b) - \Delta\xi_i + u_i^{\alpha} = 0 \\ = 0 & \text{for } i \in \mathrm{R}_0 \text{ having } -l_i(x_i^{\top} \Delta w + \Delta b) - \Delta\xi_i + u_i^{\alpha} < 0 \text{ and } i \in R_1 \\ free & \text{for } i \in V \cup E_0 \cup E_1, \end{cases} \tag{36}$$

$$z_i^{\mu} \begin{cases} \geq 0 & \text{for } i \in E_0 \text{ having } -\Delta\xi_i + u_i^{\mu} = 0 \\ = 0 & \text{for } i \in E_0 \text{ having } -\Delta\xi_i + u_i^{\mu} < 0 \text{ and } i \in E_1 \\ free & \text{for } i \in V \cup R_0 \cup R_1 \end{cases}$$

$$\text{here, } u_i^{\alpha} = -l_a \bar{w} \Delta x_a \text{ if } i = a, \text{otherwise } u_i^{\alpha} = 0$$

$$i = [1, \ldots, n]$$

Eq (36) can be further simplified with the following steps.

Noticing that for $i \in R_1$, $z_i^\alpha = 0$, and for $i \in E_1$, $z_i^\mu = 0$ and thus $z_i^\alpha = 0$. There are three groups of points with associated z_i^α being nonzero: V, R_0', E_0'. R_0', E_0' are specified below.

Denote $i \in R_0' \subset R_0$ are those having $-l_i(x_i^\top \Delta w + \Delta b) - \Delta\xi_i + u_i^\alpha = 0$, leading to $z_i^\alpha \geq 0$. For the VI, $z_i^\mu \Delta\xi_i = z_i^\alpha \Delta\xi_i = 0$ for $i \in R_0$. Therefore, $z_i^\alpha(-l_i(x_i^\top \Delta w + \Delta b) + u_i^\alpha) = 0$ for $i \in R_0'$.

The VI also states that for $i \in V$, z_i^μ and z_i^α are unrestricted, and $z_i^\mu \Delta\xi_i = 0$ and $z_i^\alpha(-l_i(x_i^\top \Delta w + \Delta b) - \Delta\xi_i + u_i^\alpha) = 0$. Therefore, $z_i^\alpha(-l_i(x_i^\top \Delta w + \Delta b) + u_i^\alpha) = 0$ for $i \in V$.

Denote $i \in E_0' \subset E_0$ are those having $\Delta\xi_i = 0$, leading to $z_i^\mu \geq 0$ and thus $z_i^\alpha \leq 0$. For the VI, $z_i^\alpha(-l_i(x_i^\top \Delta w + \Delta b) - \Delta\xi_i + u_i^\alpha) = 0$ for $i \in E_0$. Therefore, $z_i^\alpha(-l_i(x_i^\top \Delta w + \Delta b) + u_i^\alpha) = 0$ for $i \in E_0'$.

Putting these observations together, the VI is rewritten as below, which can be solved as a set of linear equations easily.

$$\Delta w = \bar{\alpha}_a l_a q + \sum_{i \in R_0' \cup V \cup E_0'} z_i^\alpha l_i x_i$$

$$\sum_{i \in R_0' \cup V \cup E_0'} z_i^\alpha l_i = 0$$

$$z_i^\alpha(-l_i(x_i^\top \Delta w + \Delta b) + u_i^\alpha) = 0 \text{ for } i \in R_0' \cup V \cup E_0'$$

$$\text{where}, u_i^\alpha = -l_a \bar{w} q \text{ if } i = a, \text{otherwise } u_i^\alpha = 0.$$

4.5. RESULTS FROM TSEP APPLICATIONS

4.5.1. LS regression-based queue length estimation

- *Queue length estimation model*

We first briefly describe the model for estimating queue length at an intersection using sampled travel time data (Ban et al., 2011). The data collected is the sampled travel times between two consecutive locations: upstream and downstream of a signalized intersection (Figure 9). Virtual

trip lines (VTL) are placed to obtain arrival times (e.g., t_{VTL1} and t_{VLT2}) while protecting data privacy. x_i^{ind} is given by t_{VTL1} and x_i^{dep} is given by $t_{VTL2} - t_{VTL1}$. The model is based on critical pattern changes in intersection travel times or delays, such as discontinuities (i.e., sudden and dramatic increases in travel times) and non-smoothness (i.e., changes of slopes of travel times), which indicate signal timing or queue length changes (Figure 10). The real-time queue length can be re-constructed by detecting these critical points in intersection travel times or delays. The model takes four steps to estimate the queue length, including data processing (computing queuing delay from the sampled travel times), cycle breaking, fitting data in each cycle $\min_{y=(w,b)} \frac{1}{2n}\sum_{i=1}^{n}(b + wx_i^{ind} - x_i^{dep})^2$ (with x_i^{ind} being the arrival (start) time at VTL$_1$ and x_i^{dep} being the travel time through the intersection), and calculating the queue length for each cycle based on the fitted line.

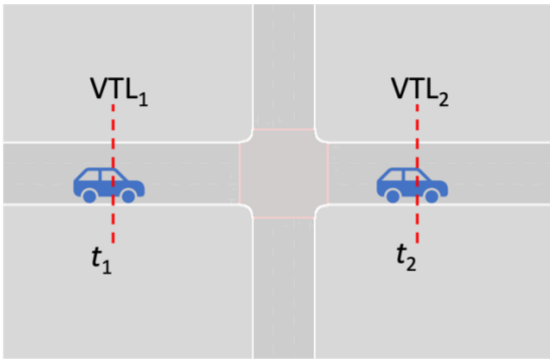

Figure 9. Collecting travel times between two locations of a signalized intersection.

Figure 11 illustrates how the model uses the measured travel time and the arrival time from mobile sensors (at the dotted line in the figure) to estimate the (piecewise) linear delay patterns of the intersection, which can be further used to compute queue length. Without introducing details that are available from the original study (Ban et al., 2009), at each cycle, the fitted queue length is proportional to the root of the fitted line: $Len_{queue} \propto -\frac{b}{w}$.

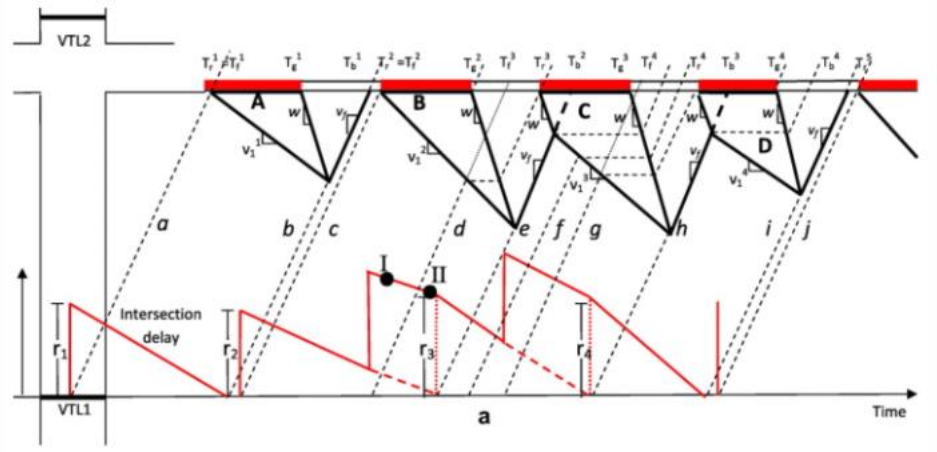

Figure 10. Intersection delay patterns (Ban et al., 2011).

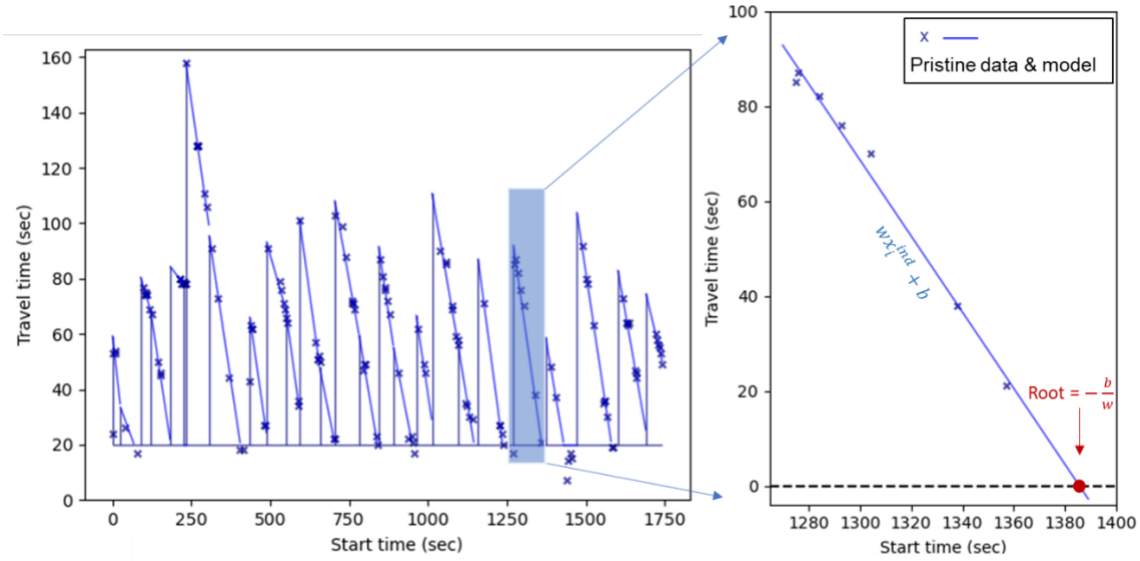

Figure 11. Illustration of fitting data in each cycle with a line to compute queue length.

- *Attack model formulation with problem-specific constraints*

Problem-specific constraints

Vehicles follow certain behaviors (e.g., car following) on the road, which could help to identify abnormal vehicles. To construct realistic and thus stealthy poisoning samples, constraints are added to express the car-following behavior at the intersection: assuming multiple vehicles travel through the intersection in a sequence, a vehicle who departs from VTL_1 first likely arrives at VTL_2 first, as VTL_1 and VTL_2 are spatially close around the intersection. This is particularly true

if the road segments have one lane at both the upstream and downstream of the intersection. If not, one can relax the constraint with a slack variable. Formally, if the targeted vehicle veh_a ($a \in (1, \dots, A)$) travels following vehicle veh_l and leading vehicle veh_f (Figure 12), denote the departure time for the three vehicles being x_l^{ind}, x_a^{ind} and x_f^{ind}, respectively. Similarly, denote the travel time from VTL_1 to VTL_2 for the three vehicles being x_l^{dep}, x_a^{dep} and x_f^{dep}, respectively, arrival time at VTL_2 can then be computed as $x_l^{ind} + x_l^{dep}$, $x_a^{ind} + x_a^{dep}$ and $x_f^{ind} + x_f^{dep}$. The constraint states as follows:

$$\begin{aligned} x_a^{ind} - x_l^{ind} &\geq hw \\ x_f^{ind} - x_a^{ind} &\geq hw \\ \left(x_a^{ind} + x_a^{dep}\right) - \left(x_l^{ind} + x_l^{dep}\right) &\geq hw \\ \left(x_f^{ind} + x_f^{dep}\right) - \left(x_a^{ind} + x_a^{dep}\right) &\geq hw \end{aligned} \tag{37}$$

Here, hw is the minimum headway.

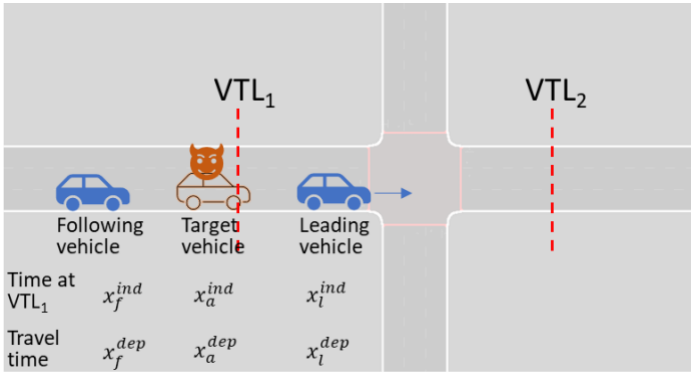

Figure 12. Illustration of car-following constraint at an intersection.

Knowledge of traffic flow dynamics can also be implemented as constraints. These constraints assure the traffic flow dynamic remains respected when crafting poisoning samples. The discharge rate of the queue (i.e., the slope of the delay pattern w) is closely related to the dynamics of traffic

shock waves, which could be known prior. Therefore, it is assumed that the range of w is known as prior knowledge, written as follows.

$$\rho_1 \leq w \leq \rho_2 \tag{38}$$

In physical meaning, the attack would be easily detected if one finds that the estimated model suggests the queue discharges either too fast or too slow.

The constraints here can certainly be extended to include more physical and/or knowledge-based constraints when constructing the attack model.

Attack model

The adversarial objective here is to maximize the deviation in queue length estimation (Eq (1)) by perturbing the collected data (i.e., arrival time x^{ind} and delay x^{dep}). Alternatively, one could specify a target queue length (e.g., doubling the length) and set the adversarial objective as inducing the estimated queue length to the target. This study tests such flexibility on the SVM-based vehicle classification model (see the next section), where a target SVM model is specified as the adversarial objective.

Denote $\hat{y} = -\frac{\hat{b}}{\hat{w}}$ as the estimation from the poisoned data and $\bar{y} = -\frac{\bar{b}}{\bar{w}}$ from the pristine data. The problem below gives the adversarial model following the constraints specified above.

$$\begin{aligned} \max_{x} \quad & O = \|\hat{y} - \bar{y}\|_2^2 \\ s.t. \quad & \left.\begin{aligned} |x - \bar{x}| &\leq \delta \\ x_l^{ind} - x_a^{ind} &\leq hw \\ x_a^{ind} - x_f^{ind} &\leq hw \\ \left(x_l^{ind} + x_l^{dep}\right) - \left(x_a^{ind} + x_a^{dep}\right) &\leq hw \\ \left(x_a^{ind} + x_a^{dep}\right) - \left(x_f^{ind} + x_f^{dep}\right) &\leq hw \end{aligned}\right\} \text{Upper level} \end{aligned} \tag{39}$$

$$\left.\begin{array}{l}(\hat{w},\hat{b}) = \underset{(w,b)}{\operatorname{argmin}} \frac{1}{2n}\sum_i \left(b + wx_i^{ind} - x_i^{dep}\right)^2 \\ \\ s.t. \quad \begin{array}{ll} g_1: & \rho_1 - w \leq 0 \\ g_2: & w - \rho_2 \leq 0 \end{array}\end{array}\right\} \text{Lower level}$$

Here, let g_1 and g_2 stand for the two constraints associated with the queue length estimation model to facilitate later discussion.

- *Lipschitz continuity*

The previous section has shown that for checking the Lipschitz continuity of the LS regression model, it only needs to validate the first sufficient condition—the linear independence constraint qualification condition, which is problem-specific. The other sufficient condition (the second order condition) has proven independent of the specific problems.

In the queue length estimation model defined above, the first sufficient condition requires that the gradients $\nabla_y g_i(\bar{x}, \bar{y})$ for all *active* constraints are linearly independent. Specifically, assuming all constraints are active, the gradients consist of

$$\begin{array}{ll} \nabla_y g_1(\bar{x}, \bar{y}) & = [-1, 0], \\ \nabla_y g_2(\bar{x}, \bar{y}) & = [1, \quad 0]. \end{array} \tag{40}$$

Serving as the upper and lower bounds of w, respectively, the two constraints will not be active at the same time. Therefore, the first sufficient condition is always satisfied and the queue length estimation model is indeed Lipschitz continuous.

- *Attack algorithm*

For the queue length estimation model, if vehicle a's data is under attack, we have

$$D_G(\bar{\boldsymbol{x}}; \Delta x_a) = \nabla_{\Delta x} O + Ds(\bar{\boldsymbol{x}}; \Delta x_a)^T \cdot \nabla_S O = \left[\frac{b}{w^2} \quad -\frac{1}{w}\right] Ds(\bar{\boldsymbol{x}}; \Delta x_a) \tag{41}$$

The semiderivative $Ds(\bar{\boldsymbol{x}}; \Delta x_a)$ is computed by

$$Ds(\bar{\boldsymbol{x}}; \Delta x_a) = \bar{s}(-B\Delta x_a), \text{ with } B = \begin{pmatrix} \nabla^2_{yx_a} L(\bar{x}_a, \bar{y}, \bar{\lambda}) \\ -\nabla_{x_a} g_1(\bar{x}_a, \bar{y}) \\ -\nabla_{x_a} g_2(\bar{x}_a, \bar{y}) \end{pmatrix}$$

Following the previous analysis and assuming the data has been normalized (i.e., $\mu = \frac{1}{n}\sum_{i=1}^{n} x_i^{ind} = 0$), we obtain the semi-derivative by solving the following problem:

$$\Delta y = \begin{bmatrix} \Delta w \\ \Delta b \end{bmatrix} = -\frac{1}{\Sigma}\left(\frac{1}{n}\begin{bmatrix} \kappa_1 \\ \kappa_2 \end{bmatrix} \Delta x + z_1 \begin{bmatrix} -1 \\ 0 \end{bmatrix} + z_2 \begin{bmatrix} 1 \\ 0 \end{bmatrix}\right), \text{ with}$$

$$z_1 \begin{cases} \geq 0 & \text{if } g_1 \text{ is active with } \bar{\lambda}_1 = 0 \text{ having } [-1,0]^\top \Delta y = 0 \\ = 0 & \text{if } g_1 \text{ is active with } \bar{\lambda}_1 = 0 \text{ having } [-1,0]^\top \Delta y < 0 \text{ OR } g_1 \text{ is inactive} \\ free & \text{if } g_1 \text{ is active with } \bar{\lambda}_1 > 0, \text{then } z_1[-1,0]^\top \Delta y = 0 \end{cases} \tag{42}$$

$$z_2 \begin{cases} \geq 0 & \text{if } g_2 \text{ is active with } \bar{\lambda}_2 = 0 \text{ having } [1,0]^\top \Delta y = 0 \\ = 0 & \text{if } g_2 \text{ is active with } \bar{\lambda}_2 = 0 \text{ having } [1,0]^\top \Delta y < 0 \text{ OR } g_2 \text{ is inactive} \\ free & \text{if } g_2 \text{ is active with } \bar{\lambda}_2 > 0, \text{then } z_2[1,0]^\top \Delta y = 0 \end{cases}$$

Here, $\Sigma = \frac{1}{n}\sum_{i=1}^{n} \left(x_i^{ind}\right)^2$, $\begin{bmatrix} \kappa_1 \\ \kappa_2 \end{bmatrix} = \frac{1}{n}\begin{bmatrix} 2\bar{w}\bar{x}_a^{ind} + \bar{b} - \bar{x}_a^{dep} & -\bar{x}_a^{ind} \\ \bar{w}\Sigma & -\Sigma \end{bmatrix}$.

The solution to Eq (42) depends on three cases that are closely related to the status of constraints at the solution pair $(\bar{x}, \bar{y}, \bar{\lambda})$.

1. No constraint is active.

The analysis for this case is simple, as $z_1 = 0$ and $z_2 = 0$.

$$\Delta y = \begin{bmatrix} \Delta w \\ \Delta b \end{bmatrix} = -\frac{1}{n\Sigma}\begin{bmatrix} \kappa_1 \\ \kappa_2 \end{bmatrix} \Delta x \tag{43}$$

Inserting it into Eq (41) leads to

$$\begin{aligned} D_G(\bar{x}; \Delta x_a) &= \left[D_G\left(\bar{x}; \Delta x_a^{ind}\right) \quad D_G\left(\bar{x}; \Delta x_a^{dep}\right)\right] = \left[\frac{\bar{b}}{\bar{w}^2} \quad -\frac{1}{\bar{w}}\right] \Delta y \\ &= \left(-\frac{\bar{b}}{n\bar{w}^2\Sigma}\left[\bar{w}\bar{x}_a^{ind} + \left(\bar{w}\bar{x}_a^{ind} + \bar{b} - \bar{x}_a^{dep}\right) \quad -\bar{x}_a^{ind}\right] + \left[\frac{1}{n} \quad -\frac{1}{n\bar{w}}\right]\right) \Delta x_a \\ &= d_{\Delta x}^1(\bar{x}) \cdot \Delta x \end{aligned} \tag{44}$$

Here, the notation $d_{\Delta x}^{1}(\bar{x})$ is introduced as the sub-gradient (the perturbation direction that can lead to the maximum impact) to facilitate our discussion later.

It can be observed that $D_G(\bar{x}; \Delta x)$ is related to some aspects of the targeted data poison $\boldsymbol{x}_a$:

- The relative position of vehicle a in the signal cycle: $\bar{x}_a^{ind}$. (Remind that the data is normalized such that $\bar{x}_a^{ind}$ here represents the relative position ($\bar{x}_a^{ind} - \mu$))
- Eq (59) suggests that selecting a point away from the middle point is likely to generate a high attack impact. In its physical meaning, attacks targeting queue head or tail lead to a higher impact than attacks on other vehicles.
- The prediction residual: $r_a = \bar{w}\bar{x}_a^{ind} + \bar{b} - \bar{x}_a^{dep}$

Eq (44) also suggests that targeting at a point associated with a large prediction residual also contributes to a high attack impact.

2. g_1 is active.

In this case, $z_2 = 0$ and (57) is simplified as:

$$\begin{array}{lll} \Delta y & = \begin{bmatrix}\Delta w \\ \Delta b\end{bmatrix} = -\frac{1}{\Sigma}\left(\begin{bmatrix}\kappa_1 \\ \kappa_2\end{bmatrix}\Delta x + z_1\begin{bmatrix}-1 \\ 0\end{bmatrix}\right) & \\ & \geq 0 \quad \text{if } \bar{\lambda}_1 = 0 \text{ and } \Delta w = 0 & \\ \text{with} & z_1\{= 0 \quad \text{if } \bar{\lambda}_1 = 0 \text{ and } \Delta w > 0 & \\ & \text{free} \quad \text{if } \bar{\lambda}_1 > 0 & \end{array} \tag{45}$$

a) If $\bar{\lambda}_1 = 0$, the solution depends on the perturbation direction Δx.

   - If the perturbation on data (i.e., x_a) Δx_a leads to $\kappa_1^{\top}\Delta x_a \geq 0$ (i.e., there exists a $z_1 \geq 0$ such that $\kappa_1^{\top}\Delta x_a - z_1 = 0$ and thus $\Delta w = 0$), the deviation of y responding to Δx is given by

$$\Delta y = \begin{bmatrix}\Delta w \\ \Delta b\end{bmatrix} = -\frac{1}{\Sigma}\begin{bmatrix}0 \\ \kappa_2\Delta x_a + z_1\mu\end{bmatrix} = \begin{bmatrix}0 \\ \frac{1}{n}[-\bar{w} \quad 1]\Delta x_a\end{bmatrix}$$

$$D_G(\bar{x}; \Delta x_a) = \left[\frac{\bar{b}}{\bar{w}^2} \quad -\frac{1}{\bar{w}}\right]\left[\frac{0}{\frac{1}{n}[-\bar{w} \quad 1]\Delta x_a}\right] = \frac{-1}{n\bar{w}}[\bar{b} \quad 1]\Delta x_a = d^2_{\Delta x}(\bar{x}) \cdot \Delta x_a \qquad (46)$$

Here, we introduce a notation $d^2_{\Delta x}(\bar{x})$ for convinience of discussion.

- If Δx_a leads to $\kappa_1^\top \Delta x_a < 0$, with $z_1 = 0$ (because $\Delta w = \frac{-1}{\Sigma}\kappa_1^\top \Delta x_a > 0$),

  $D_G(\bar{x}; \Delta x_a) = d^1_{\Delta x}(\bar{x}) \cdot \Delta x_a$ following Eq (44).

b) If $\bar{\lambda}_1 > 0$, the solution needs to meet $z_1[-1,0]^\top \Delta y = 0$. Since z_1 is free, this means either $z_1 = 0$ or $\Delta w = 0$.

If $z_1 = 0$, $D_G(\bar{x}; \Delta x_a) = d^1_{\Delta x}(\bar{x}) \cdot \Delta x_a$ following Eq (44).

If $\Delta w = 0$, $D_G(\bar{x}; \Delta x_a) = d^2_{\Delta x}(\bar{x}) \cdot \Delta x_a$ following Eq (46).

In summary, if g_1 is active, there are two groups of semi-derivatives depending on Δx_a. And $D_G(\bar{x}; \Delta x_a)$ can be either computed by $d^1_{\Delta x}(\bar{x}) \cdot \Delta x_a$ or $d^2_{\Delta x}(\bar{x}) \cdot \Delta x_a$. When implementing Algorithm 1, the impacts of two directions (i.e., $\frac{d^1_{\Delta x}(\bar{x})}{|d^1_{\Delta x}(\bar{x})|}$ and $\frac{d^2_{\Delta x}(\bar{x})}{|d^2_{\Delta x}(\bar{x})|}$) is compared and the one with a larger impact is selected as the perturbation direction.

3. g_2 is active.

In this case, $z_1 = 0$. The analysis is similar to the previous case and the same (two) groups of semi-derivatives are obtained. Details are omitted here.

Note that the only difference with the previous case is that when $\bar{\lambda}_2 = 0$, $d^2_{\Delta x}(\bar{x})$ is obtained when $\kappa_1^\top \Delta x_a \leq 0$ instead of $\kappa_1^\top \Delta x_a \geq 0$.

Notice that Eq (43) (i.e., the result with no constraint is active) is consistent with the gradient computed in the previous study of regression models free of constraints (Jagielski et al., 2018). The results show that the constraints play a role in evaluating the response of the TSEP solution to data changes. This suggests that the existing gradient-based attack methods may fail without considering the constraints, especially when an inequality constraint is active.

- *Effectiveness of attack model*

We first demonstrate the effectiveness of our attack model by sequentially perturbing the data points. The proposed attack in Algorithm 1 is compared with the state-of-the-art gradient-based attack. For convenience, the two are called the semi-derivative and gradient methods, respectively. The gradient method applied in the previous studies is different from Algorithm 1 in two aspects: 1) it randomly selects a point as the target; 2) it perturbs the target point following the gradient. Additionally, for a more fair comparison, we improve the gradient method so that the implementation is in line with Algorithm 1. Specifically, the improved gradient method takes a similar data selection procedure and selects the target point based on the gradients computed for all data points. Then, the only difference is the way to apply perturbation (i.e., following gradients instead of semi-derivative).

Figure 13A shows the convergence of the adversarial objective following semi-derivative and gradient methods. In general, our attack steadily increases the objective value (i.e., the deviation of estimated queue length) faster than the gradient method. It is surprising to observe that without considering the effects of constraints, keeping perturbing the data point following the gradient methods (including the improved one) would even reduce the objective. These observations demonstrate the advantages of our attack method. The convergence rate of the attack methods reduces starting at the 12^{th} iteration, as constraints become active. This can be clearly observed in Figure 13B, which shows that the slope of fitted line w reaches the upper bound at the 12^{th} iteration.

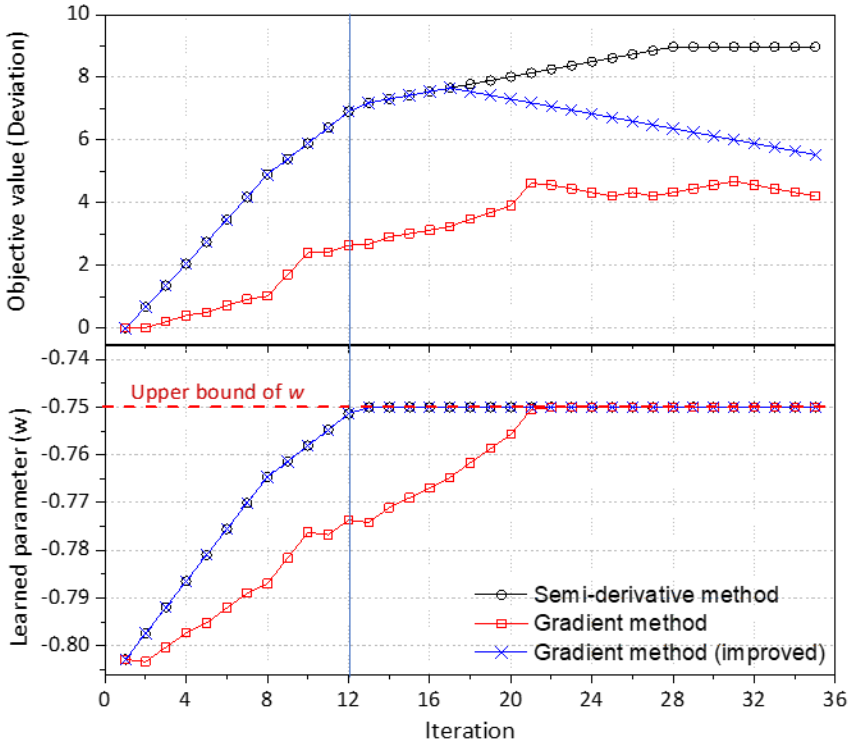

Figure 13. A) Convergence of adversarial objective when perturbing queue length estimation model; B) Changes in the learned parameter (w) following our attack.

To help understand the advantages and the features of our attack, Figure 14 compares the directions of gradients and max-impact semi-derivative (Δx_i^*, see Algorithm 1) together with their impacts (assuming a unit perturbation). The arrows indicate the directions at each point and the arrow lengths are proportional to the impacts. Figure 14 presents results from the beginning of attacks and after w hits the upper bound (i.e., constraint g_1 is active). It can be observed that before a constraint becoming active, the directions of gradients and max-impact semi-derivative are consistent. The directives are intuitive to understand: the slope is to be rotated counterclockwise to deviate the root of the fitted line, which is achieved by perturbing points around the queue head "downward" while those around the queue tail "upward". Meanwhile, attacking points (vehicles) at the head or the tail of the queue leads to larger impacts than attacking those in the middle, and clearly, point 6 is targeted at the beginning of the attack.

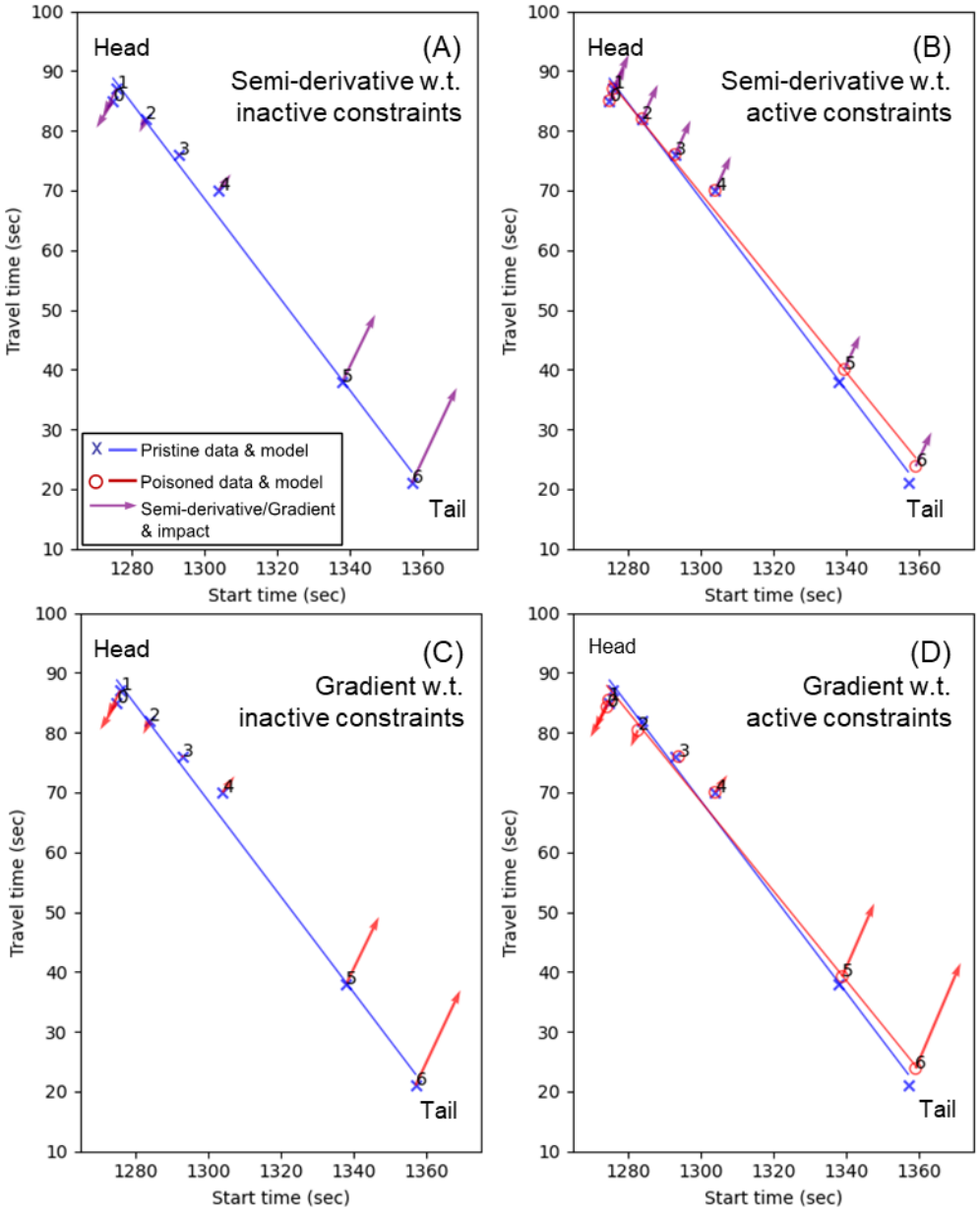

Figure 14. Max-impact semi-derivative Δx_i^* and the impact $D_G^*(x; \Delta x_i)$ at each point (A) when the constraint is inactive and (B) when the constraint is active; Gradient and the impact at each point (C) when the constraint is inactive and (D) when the constraint is active.

However, semi-derivatives change after the constraint becomes active: they are now in the same (upward) direction and tend to induce the same impact. In contrast, the gradients are still the same as if the constraint is inactive. These semi-derivatives seem counterintuitive but can be explained with some examples. First, assuming point 0 is perturbed downwards following the

gradient's direction, the fitted line can no longer be rotated counterclockwise due to the constraint. In order to reduce the loss caused by the perturbation, the fitted line will keep the slope unchanged but shift towards the left, which decreases the root and drops the objective value. This explains why in Figure 13A the objective value following the gradient method cannot increase reliably.

In contrast, if point 0 is perturbed upward following the semi-derivative, the fitted line has a chance to increase the root by staying at the same slope and slightly shifting towards the right. Intuitively, increasing the root by such a parallel shift can be less effective than rotating the line (before constraints become active). This explains why the convergence rate of our attack drops after the 12^{th} iteration (Figure 13A).

Test stealthiness of the attack: In order to validate the stealthiness of the attack, existing defense methods are implemented and tested on the poisoned datasets, including Huber regression, RANSAC and TRIM algorithm. It is found that these existing defense methods could not effectively mitigate the negative effects (i.e., deviation of queue length estimates). The details of the tests and results are reported in Section 5.2, where these existing defense methods are compared with the proposed IED solution, a novel and effective defense.

4.5.2. SVM-based vehicle classification

Vehicle classification is important for traffic management, operation, and transportation planning. This study investigates an SVM-based method that classifies vehicles into trucks and passenger vehicles using GPS data extracted from mobile traffic sensors. SVMs are among the best (and many believe are indeed the best) "off-the-shelf" supervised learning algorithms. Therefore, investigating the vulnerability of the SVM-based TSEP model could generate implications beyond the transportation community.

- *SVM-based vehicle classification model*

Without introducing the details in the original study (Sun and Ban, 2013), the model can be briefly summarized in the following four steps:

a) Collect GPS data of passenger cars and trucks.

b) Generate speed and acceleration-/deceleration-related features for each vehicle using GPS data (Figure 15). The step for feature selection in the original study is omitted here. This study directly takes the conclusion—the variations of accelerations and decelerations are selected as the input of the SVM model.

c) Train an SVM classifier for vehicle classification.

d) Apply the trained SVM classifier to the test dataset.

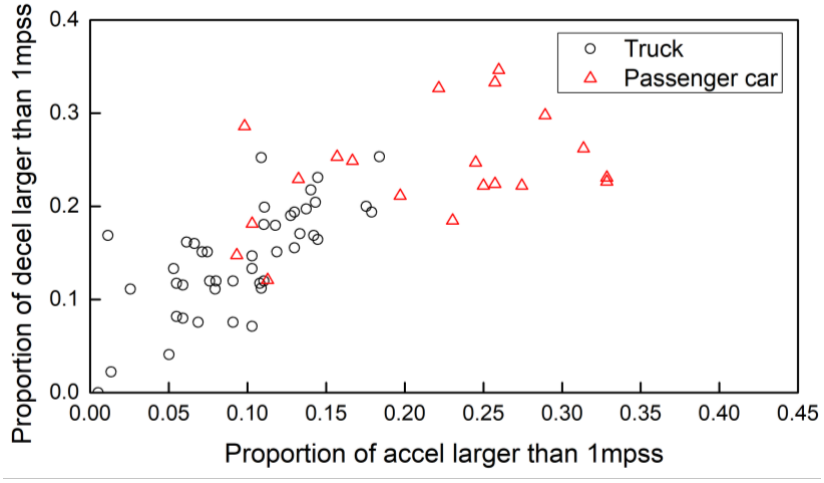

Figure 15. Data for vehicle classification.

The TSEP model is written as:

$$\begin{aligned} \min_{y=(\boldsymbol{w},b,\xi)} \quad & \frac{1}{2} \| \boldsymbol{w} \|_2^2 + C \sum_i^n \xi_i \\ s.t. \quad & g_i^{\alpha} = 1 - \xi_i - l_i(\boldsymbol{w}\boldsymbol{x}_i + b) \leq 0 \\ & g_i^{\mu} = -\xi_i \leq 0 \\ & i = 1, \dots, n \end{aligned} \tag{47}$$

Here, $\boldsymbol{x}_i$ gives (two) feature values extracted from the trajectory of vehicle i, and l_i is the binary label for the vehicle (i.e., $l_i = 1$ for trucks and $l_i = -1$ for passenger vehicles). The hyperplane is a vector of length two (i.e., $\boldsymbol{w} = [w_1, w_2]$). C is a constant associated with the penalty term allowing for a soft margin.

Figure 16 shows the SVM boundary learned from the original data, which gives $\boldsymbol{w} = [-11.2, -2.5]$ and $b = 6.7$. It can observed that the decision boundary weighs heavily on w_1, meaning that acceleration acts as the dominant feature for grouping the two vehicle classes.

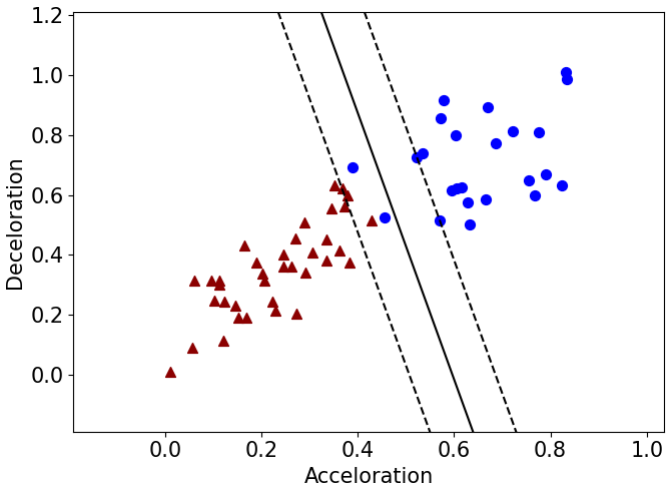

Figure 16. SVM learned from the clean data

- Attack model formulation with problem-specific constraints

Problem-specific constraints

The SVM model with slack variables contains inherent constraints (Equation (47)). These constraints complicate the mapping from the data perturbation and TSEP solution, as the constraints themselves are subject to changes due to the data perturbation.

Besides SVM's inherent constraints, the feature values in the data are confined by practical limits such as vehicle movements and dynamics. Specifically, the feature values for vehicle classification are closely related to patterns of vehicle movement (e.g., speed) and dynamics (e.g.,

acceleration and deceleration). Given that only the variations of accelerations $\boldsymbol{x}^{ac}$ and decelerations $\boldsymbol{x}^{dc}$ are selected as the input, constraints are added by specifying the minimum and maximum of $\boldsymbol{x}^{ac}$ and $\boldsymbol{x}^{dc}$ (Eq (48)), which are related to vehicle design and driving conditions. This study identifies the bounds from the pristine dataset as the minimum and maximum feature values.

$$\begin{aligned} \rho_1 \le x_i^{ac} \le \rho_2 \\ \rho_3 \le x_i^{dc} \le \rho_4 \end{aligned} \tag{48}$$

Attack model

The adversarial targets at misleading the SVM model into believing that the two features (i.e., acceleration and deceleration) are equally important for vehicle classification. As such, a model-target poisoning attack is formulated, where the objective is to induce the model parameter $\boldsymbol{w}$ towards the adversarial goal $\boldsymbol{w}^* = (w_1^*, w_2^*)$. Here, targeting a model with two equally important feature weights, let $|w_1^* - w_2^*| = 0$, which can be induced by minimizing $|w_1 - w_2| - |w_1^* - w_2^*| = |w_1 - w_2|$. Specifically, the attack model can be written as follows:

$$\begin{aligned} &\min_{\boldsymbol{x}} \quad O = |\widehat{w}_1 - \widehat{w}_2| \\ &\left.\begin{aligned} \textit{s.t.} \quad & |x - \bar{x}| \le \delta \\ & \rho_1 \le x_i^{ac} \le \rho_2 \\ & \rho_3 \le x_i^{dc} \le \rho_4 \end{aligned}\right\} \text{Upper level} \\ &\left.\begin{aligned} & (\widehat{w}_1, \widehat{w}_2) \in \underset{y=(\boldsymbol{w},b,\boldsymbol{\xi})}{\arg\min} \tfrac{1}{2} \| \boldsymbol{w} \|_2^2 + C \textstyle\sum_i^n \xi_i \\ \textit{s.t.} \quad & g_i^{\alpha} = 1 - \xi_i - l_i(w_1 x_i^{ac} + w_2 x_i^{dc} + b) \le 0 \\ & g_i^{\mu} = -\xi_i \le 0 \end{aligned}\right\} \text{Lower level} \\ &\quad i = 1, \dots, n \end{aligned} \tag{49}$$

Notice that in attacking a classification model, perturbed labels are often not included as part of decision variables, as labels are discrete values and including them will lead to a label-flipping

problem that would lead to a different attack type (Mei and Zhu, 2015). Therefore, different from attacking regression models, only the feature values $\boldsymbol{x}_i$ are perturbed.

- *Lipschitz continuity*

Following the analysis in the previous section, the (two) sufficient conditions are checked to ensure that $\bar{S}$ *has a Lipschitz continuous single-valued* $\bar{s}$ *around (0, 0) for (0, 0)*. Since the previous analysis has shown that $\frac{1}{2}\langle \Delta y, \nabla_{yy}^2 L(\bar{x}, \bar{y}, \bar{\alpha}, \bar{\mu}) \Delta y \rangle > 0$ for every nonzero $\Delta y \in M^+$, it only needs to validate that the gradients $\nabla_y g_i(\bar{x}, \bar{y})$ for all active constraints are linearly independent.

Notice the gradient vectors of interest are

$$\begin{aligned} \nabla_y g_i^{\alpha}(\bar{x}, \bar{y}) = \quad & [-l_i x_{i,1} \quad -l_i x_{i,2} \quad -l_i \quad 0 \quad \dots \quad -1 \quad \dots \quad 0] \quad \text{for } i \in R_0 \cup V \cup E_0 \cup E_1 \\ \nabla_y g_i^{\mu}(\bar{x}, \bar{y}) = \quad & [\quad 0 \quad\quad 0 \quad\quad 0 \quad 0 \quad \dots \quad -1 \quad \dots \quad 0] \quad \text{for } i \in E_0 \cup V \cup R_0 \cup R_1 \end{aligned}$$

Our numerical test shows that the vectors are indeed linearly independent of the vehicle classification dataset. Specifically, we perform *LU* decomposition of the matrix consisting of these vectors and inspect the returned *U* matrix, finding no null pivot vector.

Therefore, for our SVM-based vehicle classification problem, the mapping S has a Lipschitz continuous single-valued localization s around $\bar{x}$ for $(\bar{y}, \bar{\alpha}, \bar{\mu})$ and this localization s is semi-differentiable at $\bar{x}$.

- *Attack algorithm*

Assuming data of vehicle a is under attack, we determine how the adversarial objective responds to the perturbations Δx_a that is applied to data $\bar{x}_a$.

$$D_G(\bar{x}; \Delta x_a) = 2(\bar{w}_1 - \bar{w}_2)[1 \quad -1] \cdot \Delta w \tag{50}$$

Following the analysis earlier, Δw is obtained from the semiderivative $D_S(\bar{x}; \Delta x_a)$ and is computed by solving Equation (42) with $-B\Delta x_a$ as its input. It is found that the results depend on which set the target point a belongs.

- *Effectiveness of attack*

We first demonstrate the effectiveness of our attack (the semi-derivative method following Algorithm 1) by sequentially perturbing the data points. Figure 17 compares the convergence to the adversarial objective following semi-derivative and gradient methods. Our attack steadily reduces the difference and consequently converges to the target model. As shown in Figure 17B, six data points are slightly perturbed at the convergence while the SVM's decision boundary is significantly shifted, making the two features equally important to separate cars and trucks. This study also tests the effect of perturbing multiple (three) points simultaneously at each iteration. Figure 17A suggests that the convergence can significantly speed up.

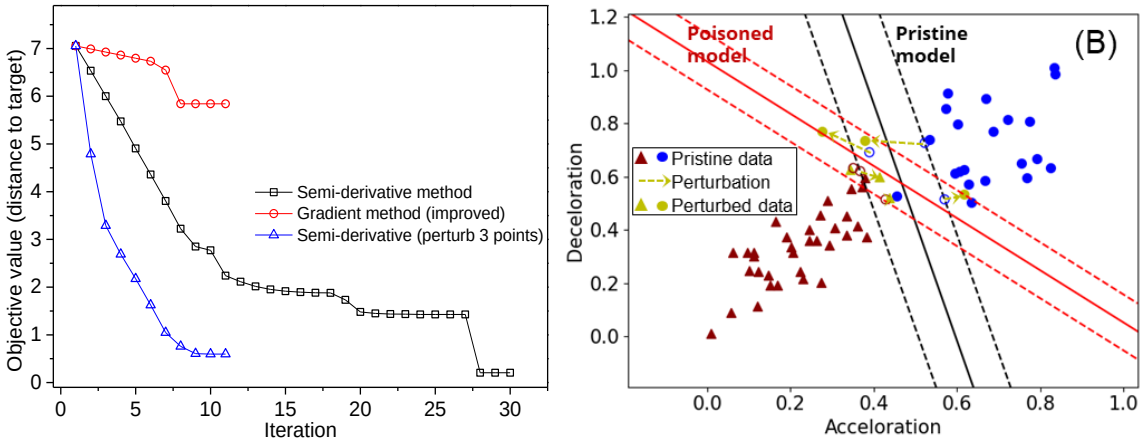

Figure 17. A) Comparison of convergence of gradient and semi-derivative method. B) The poisoned model and the perturbed data following our attack.

In contrast, the improved gradient attack does not seem to converge toward the target model reliably (Figure 17A). Concretely, the gradient attack not only converges at a much slower rate than the semi-derivative method but also stops making further progress after a few iterations. Here, the (origianl) gradient method that attacks a randomly selected point is not implemented, as the random selection tends to pick up a non-support vector, which does not respond to the perturbation.

The following Investigation tries to further understand the differences of the semi-derivative and gradient methods in their convergence behaviors.

Test stealthiness of the attack: In order to validate the stealthiness of the attack, common attack detectors are implemented to check whether the attack could be detected. Specifically, anomaly detectors (e.g., OCSVM and Isolation Forest) are trained as attack detectors using the clean datasets. Then the detectors are implemented to detect poisoned data poisons in the poisoned dataset. It is found that the poisoned data poisons cannot be detected, suggesting the stealthiness of the proposed data poisoning attacks.

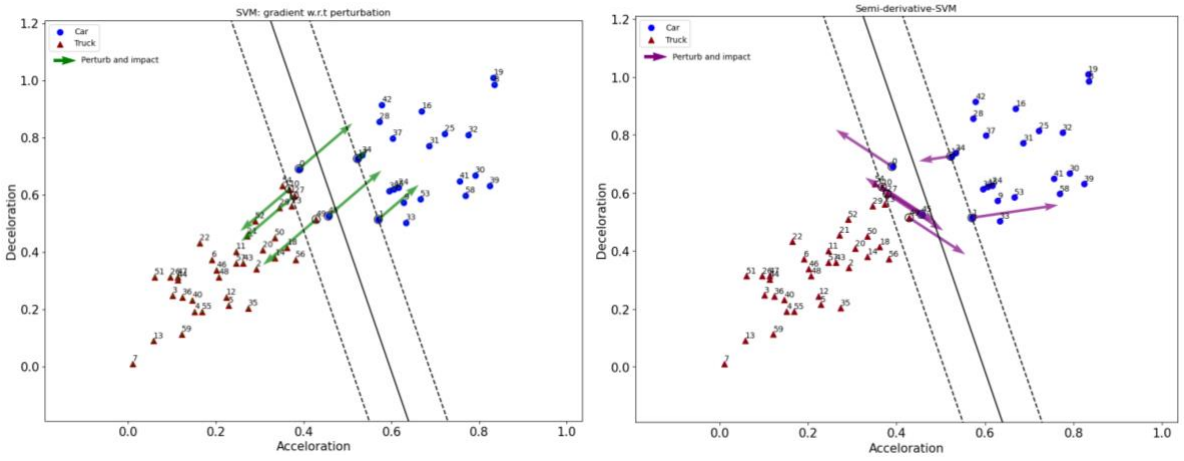

Figure 18. A) Gradient and its impact at each data point; B) Max-impact semi-derivative and its impact.

Figure 18 compares the directions of gradients and semi-derivatives (Δx_i^*, see Algorithm 1) together with their impacts assuming a unit perturbation. It can be observed that only the gradients of support vectors are nonzero, which makes sense. Additionally, these gradients are in parallel with each other, while those associated with the two classes are in opposite directions. To understand this, we simplify the computation of gradients and find that the gradient at data x_a is determined by

$$\nabla_{x_a} g = \bar{\alpha}_a \cdot l_a \cdot \Lambda_S \tag{51}$$

See details of obtaining Eq (51) in the appendix. Here Λ_S is a vector relying on the data in group S only. Also, remind that l_a gives the binary label and $\bar{\alpha}_a$ is the dual variable, based on which data points are grouped into margin support vectors ($0 < \bar{\alpha}_a < C$, set V), error support vectors ($\bar{\alpha}_a = C$, set E) and reserve points ($\bar{\alpha}_a = 0$, set R). Eq (51) explains the previous observations. Additionally, it suggests (also observable from Figure 18A) that the impacts of perturbing margin support vectors are smaller than perturbing error support vectors. Such observation is counterintuitive, as one would expect the margin support vectors would generally be more sensitive to perturbations. It can be shown later that this could be a misleading conclusion, which also explains why the convergence rate of the gradient method is slower than the semi-derivative, as the former perturbs an error support vector in each iteration. Moreover, Eq (51) suggests that the gradient method relies on a non-empty data set of V to identify the gradients, as the method needs a set of equations associated with set V from KKT conditions. If S does go empty, the gradient method will fail to proceed, as shown in Figure 17A.

The max-impact semi-derivative and its impact at each data point are shown Figure 18B. The results show some similarities but many significant differences with the gradients in Figure 18A. Regarding the similarities, there is no impact by perturbing any point in R following the semi-derivative method. And the directions of the semi-derivatives in E are in parallel with each other while those at the two classes of points are in opposite directions. However, it can be observed that these max-impact semi-derivatives in E are distinct from the gradients. A close investigation (see appendix) suggests that the semi-derivatives in E are given by $C \cdot l_a \cdot \Omega(\Delta x)$, which explains these observations. Here, $\Omega(\Delta x)$ represents a vector computed from Δx. Moreover, the semi-derivatives at data points in set V can be different from each other and seem not relevant to those in E. And

the impact of perturbing a point in V (e.g., point 1) can be larger than perturbing any point in E, in contrast to the observation from the gradient method above.

Adding poisoning data: Compare with model target-based attack

We also compare our attack with the state-of-the-art model-targeted attack (the *MaxLoss* method). *MaxLoss* can effectively induce the SVM model towards a pre-specified target by sequentially adding poisoning data. At each iteration, it adds the poisoning point with the maximum loss difference between the intermediate model obtained and the target model. It was shown that following the iterations, the attack gradually minimizes the maximum loss difference between the induced intermediate model and the target model, eventually obtaining a model similar enough to the target model.

To be comparable with the *MaxLoss* method, Algorithm 1 is modified to add new poisons (see appendix for Algorithm 2). Specifically, a new point is initialized by duplicating a (randomly selected) pristine point near the margin lines. Then it is perturbed following the computed semi-derivative until certain conditions/constraints are met before adding another new poison. The algorithm stops when the convergence condition is satisfied.

Figure 19 shows that the semi-derivative method via adding new poisons can effectively induce the model to the target after adding nine new poisons. Figure 20A shows the new poisons and the poisoned model. It can be observed that the attack could be stealthy as the new poisons are close to the pristine data.

Figure 19 also validates that the *MaxLoss* method is effective: the target model is induced by adding a few (four) points. However, as shown in Figure 20B, the new poisons generated by the MaxLoss method are not stealthy. The new points are generated at the boundary limiting the feature values.

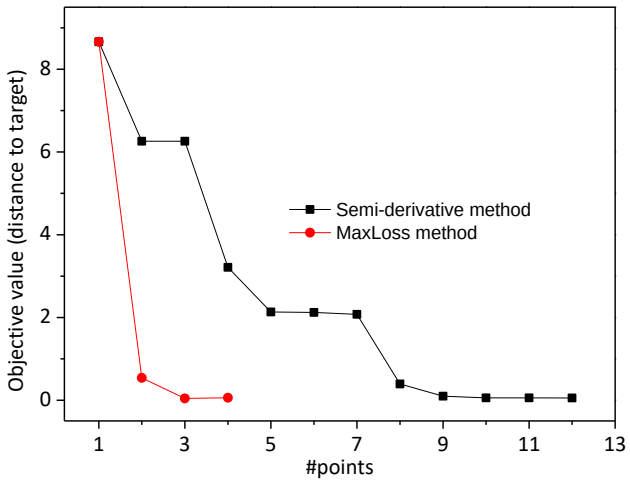

Figure 19. Convergence to the target SVM model by adding new poisons: comparing our attack with MaxLoss method.

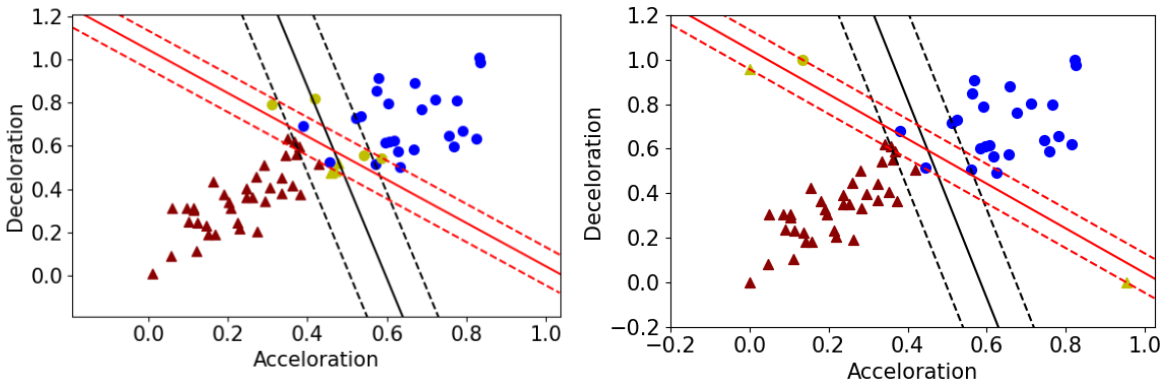

Figure 20. Results of adding new poisons: A) poisons added using our attack; B) poisons added using MaxLoss method.

4.6. SUMMARY

This work formulates data poisoning attacks against TSEP models as a general sensitivity analysis of optimization problems over data perturbations (attacks) and studies the Lipschitz continuity property of the TSEP solutions. A generalized implicit function theorem for the problem is introduced, enabling attack models to fit a broader spectrum of TSEP applications, especially those with general constraints. The proposed method is applied and tested in widely used (real-world)

TSEP models. The results show that the proposed method can outperform the state-of-the-art gradient method in terms of flexibility and the effectiveness of attacks. It is demonstrated that without considering the role of constraints, the gradient methods would be slow in moving toward the adversarial objectives or even fail to advance when the constraints become active. The study has multiple implications, including promoting the understanding of the vulnerability of data-driven transportation systems and helping to develop effective defense methods.

5. INFRASTRUCTURE-ENABLED DEFENSE SOLUTIONS

This chapter develops infrastructure-enabled solutions against data poisoning attacks in transportation, starting with vehicular data and then data for TSEP applications. The first part of this chapter presents the work on defending against attacks on vehicular data, using GPS spoofing as the case study to illustrate the main idea. The design of infrastructure-enabled GPS spoofing detection and correction is presented, followed by preliminary results from experimental tests using both simulation and real-world GPS data, under various scenario settings and different spoofing attacks. The effectiveness of the infrastructure-enabled solution is evaluated by comparing it with state-of-the-art GPS spoofing defense strategies. The second part presents the work defending against attacks on TSEP models, using the queue length estimation model as the case study. It shows how the secure infrastructure data (e.g., signal status and loop detector data) can help detect attacks on mobile travel time data, which has been shown vulnerable to data poisoning attacks in the previous chapter. A framework for detecting and correcting attacks is designed, followed by simulation tests to demonstrate its effectiveness.

5.1. INFRASTRUCTURE-ENABLED DEFENSE METHOD AGAINST GPS SPOOFING

5.1.1. Problem Statement

Though a common means to acquire a vehicle's location, GPS updates at a relatively low frequency (often one second or longer) and could be out of service due to multiple factors (e.g., traveling in an underground tunnel). Modern localization solutions often incorporate the vehicle motion model to track vehicle locations at a higher frequency, especially in applications requiring high-frequency location updates and continuity-of-service (e.g., autopilot in advanced driving systems). The

motion model requires highly frequent local measurements as the input. Yet, due to noises or errors in local measurements, the motion model needs correction periodically using global measurements.

We consider a simple yet common localization solution: a vehicle can be tracked by a typical motion model with local measurements (e.g., accelerometers and gyroscope) from a low-end IMU as the input and takes global measurements from GPS for correcting location errors periodically. Low-end IMUs are pervasive nowadays and are widely deployed in smartphones and vehicles. The problem setting here ensures the generality of the study since one can obtain IMU measurements from a vehicle's OBD portal (Boriboonsomsin et al., 2010), without installing additional sensors or utilizing the data from such sensors even if they are installed (note that these sensors themselves could be attacked). Furthermore, IMU measurements are safe from malicious attacks (especially remote attacks), as they are not collected by interacting with the potentially adversarial environment (Petit and Shladover, 2015). GPS measurements however can be spoofed, as discussed above, resulting in falsified measurements. Without detecting the spoofed GPS measurements, the vehicle could deviate from the desired trajectory, leading to possibly disastrous consequences. Our goal here is to propose an infrastructure-enabled solution with which the vehicle can utilize the secure measurements from RSUs to timely detect GPS spoofing and correct location errors resulting from the attacks, as illustrated in Figure 1. Section 5.1.3 provides more details about the data provided by the RSU.

In this section, we first briefly introduce a typical vehicle motion model based on low-end IMU measurements and the mathematical model of GPS measurements. We then describe a typical, widely used Kalman Filter (KF)-based localization method that fuses local and global measurements.

5.1.2. Preliminaries

- *Local Measurements and Vehicle Motion Model*

Given a vehicle's initial position, the motion model tracks the vehicle using local measurements (e.g., from IMU) at a high frequency. Unlike global measurements that directly provide vehicles' locations in the global coordinate frame, these local measurements provide information regarding vehicles' velocities and changes in heading from time to time in the local coordinates frame (also termed as the body frame in the literature). A typical vehicle motion model is illustrated in Figure 21: with local measurements (velocity and steering angle), the vehicle tracks its movement from the previous location x_{k-1} (in black) to a new location x_k (in blue). Local measurements from different sensors vary in their forms, such as accelerometer and gyroscope from an IMU or steering angle and odometry readings from a wheel encoder. As noted above, this study focuses on IMU measurements.

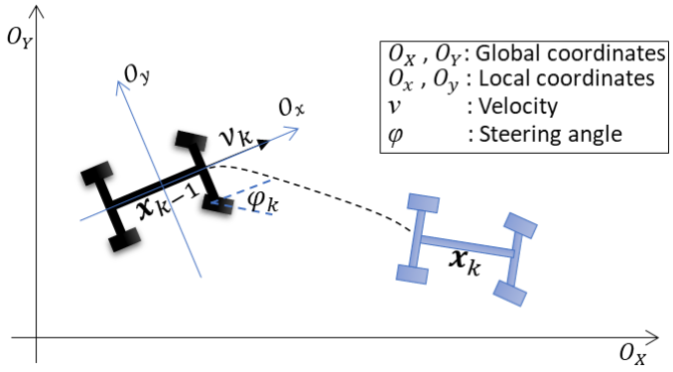

Figure 21. Tracing vehicle location via the vehicle motion model.

Given local measurements as the input, a discretized vehicle motion model can be written in a nonlinear differential equation as follows (Dissanayake et al., 2001; Skog and Handel, 2009).

$$\boldsymbol{x}_k = f(\boldsymbol{x}_{k-1}, \boldsymbol{u}_k) \tag{52}$$

Here, $\boldsymbol{x}_k$ represents the vehicle position at time step k in the global coordinate frame, and $\boldsymbol{u}_k$ represents the local measurement. A detailed discussion and implementation of Equation (52)

based on low-end IMU can be found in Skog and Handel (2009), and omitted here. By iterating Equation (52), the vehicle position can be perfectly tracked if there are no or negligible errors in the local measurements (e.g., measurements from high-end IMUs). In reality, however, the local measurements collected from onboard sensors (e.g., low-end IMUs) often suffer from errors or noises. Accordingly, Equation (52) is modified as follows.

$$\boldsymbol{x}_k = f(\boldsymbol{x}_{k-1}, \boldsymbol{u}'_k) = f(\boldsymbol{x}_{k-1}, \boldsymbol{u}_k + \boldsymbol{w}_k) \quad (53)$$

Here, $\boldsymbol{u}'_k$ and $\boldsymbol{w}_k$ represent the noisy local measurement and additive white noise with a covariance matrix $\boldsymbol{Q}$, respectively. As a result, errors associated with the estimated position may accumulate over time due to the integrative nature of the motion model. Therefore, the estimate errors need to be periodically corrected using global measurements (e.g., from GPS).

- *Global Measurements from GPS*

Global measurements of vehicle positions from various sensors can be used to correct vehicle motion model errors accumulated over time (Gao et al., 2015). Among these sensors, GPS is often combined with low-end IMU to correct errors accumulated from IMU noises. It provides global yet low-frequent (e.g., 1Hz) measurements (e.g., vehicle location and velocity). Mathematically, GPS measurements, $\boldsymbol{z}_k^{GPS}$, are modeled in Equation (54) (Wang et al., 2021b).

$$\boldsymbol{z}_k^{GPS} = \boldsymbol{H} \times \boldsymbol{x}_k + \boldsymbol{e}_k^{GPS} \quad (54)$$

Here $\boldsymbol{H}$ is a matrix mapping vehicle position to the measurement space. $\boldsymbol{e}_k^{GPS}$ is the measurement noise which is assumed to be additive white noise with covariance matrix $\boldsymbol{R}^{GPS}$.

- *EKF-based Localization Model*

Estimating vehicle positions from multiple sensor measurements can be achieved by a KF-based method or its variants (Schreiber et al., 2016; Toledo-Moreo and Zamora-Izquierdo, 2010). Here we briefly describe the KF-based localization model used in this dissertation (Figure 22) to

combine GPS (global) and IMU (local) data, i.e., to periodically correct IMU errors using GPS data. Vehicle (global) location at time k is represented by the KF's state $\hat{\boldsymbol{x}}_k$ and state uncertainty with a covariance matrix $\hat{\boldsymbol{P}}_k$. Due to the non-linearity of the vehicle motion model (Equation (52)), we adopt an Extended Kalman Filter (EKF) proposed in Shen et al. (2020), which is a typical localization algorithm to fuse IMU and GPS measurements.

Following initialization at $k = 0$, EKF estimates the vehicle positions by iterating a prediction step and an update step. The prediction step iterates the motion model (Equation (53)) to predict the vehicle positions at the IMU frequency; the process is often called *dead-reckoning*. The update step runs at the frequency of GPS measurements to correct errors accumulated in the prediction step.

Note that a vehicle relies on EKF for location estimation only when GPS is not spoofed. If a spoofing attack is detected, the EKF iteration will be interrupted as GPS measurements are discarded. In this case, RSU data will be used to estimate vehicle locations together with the IMU data; see Section 5.1.3.

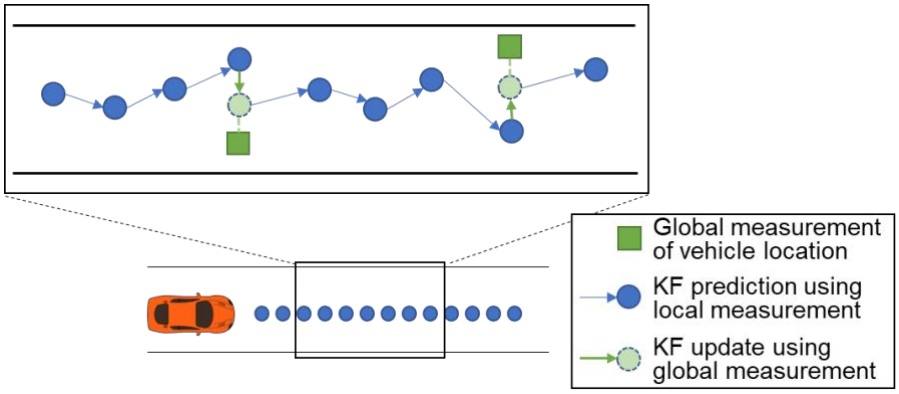

Figure 22. Illustration of KF-based localization solution

- *Attack Models*

Attack models are essential for the investigation of attack detection and mitigation. This study considers two types of GPS spoofing attacks: the constant bias attack and the stealthy attack. Other types of attacks on GPS discussed in Chapter 2 are not implemented in this study, since they either fall out of the scope of this study (e.g., DoS attacks) or can be approximated by the constant bias or stealthy attacks (van Wyk et al., 2020).

Constant Bias Attack: A constant bias attack injects a constant bias into the true measurements, temporarily causing the GPS readings to deviate from the true ones. In practice, attackers could launch a bias attack to mislead a vehicle by adding a lateral offset or a longitudinal offset (or both) to the true GPS readings $\boldsymbol{z}_k^{GPS}$. Mathematically, the received GPS measurement would be

$$\tilde{\boldsymbol{z}}_k^{GPS} = \boldsymbol{z}_k^{GPS} + \boldsymbol{C}\ (k \in [t_s, t_e]), \tag{55}$$

where $\tilde{\boldsymbol{z}}_k^{GPS}$ is the spoofed GPS data, and $\boldsymbol{C}$ is a constant vector that can be added to the true GPS readings. t_s and t_e represent the start time and end time of the attack, respectively. With a constant bias attack, the vehicle may be deceived by believing that it is on the wrong lane or a wrong location on the roadway and thus takes faulty actions.

Stealthy Attack: A stealthy attack injects a sequence of increasing deviations into the true measurements, such that the vehicle gradually drifts away from its true trajectory. Mathematically, the received GPS measurement can be expressed as:

$$\tilde{\boldsymbol{z}}_k^{GPS} = \boldsymbol{z}_k^{GPS} + \boldsymbol{c}_k\ (k \in [t_s, t_e]), \tag{56}$$

where $\boldsymbol{c}_k$ is carefully designed to avoid triggering an attack detector. Stealthy attacks are more deceptive than constant bias attacks; multiple such strategies have been proposed for GPS spoofing. As noted in Chapter 2, we implement FusionRipper. This state-of-the-art GPS spoofing strategy is recognized by top-tier cybersecurity communities and can effectively fail production-grade automated driving systems (Shen et al., 2020b). In this study, the implementation of FusionRipper

is simplified, since our localization solution includes no LiDAR as in the original study. Specifically, we skip the vulnerability profiling step and implement the aggressive spoofing step directly. The aggressive spoofing performs exponential spoofing that increases the deviation $\boldsymbol{c}_k$ exponentially. As shown by Equation (57), $\boldsymbol{c}_k$ is controlled by two parameters: *m* and *n*. Figure 23 illustrates the deviation as a function of time (The setting *m* = 0.7 and *n* =1.1 follows FusionRipper in the original paper). It can be observed that at the beginning of the attack, the deviation is small, making it difficult to be detected. As a result, the spoofed GPS measurements would be fused and corrupt the data fusion framework (i.e., EKF in our study). Once this occurs, aggressive deviations can be injected without alerting the detection algorithm.

$$\boldsymbol{c}_k = m * n^k \tag{57}$$

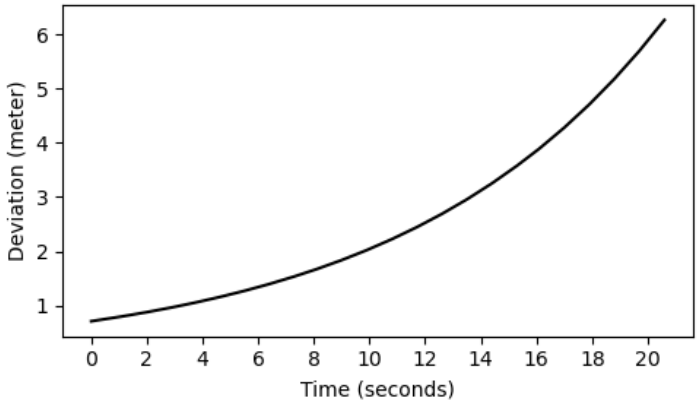

Figure 23. Illustration of aggressive spoofing in the stealthy attack.

5.1.3. Methodology

The overall workflow of the proposed infrastructure-enabled defense method is illustrated in Figure 24. It includes three components. *The first component* in the infrastructure-enabled defense method aims to obtain secure, global measurements of vehicle positions from RSUs. The design of secure RSU data ensures that a vehicle can use RSU data to obtain a global position measurement of the vehicle (like GPS measurement). Another essential part is to secure the RSU

data. This, again, is an extensively studied area in the cybersecurity field. This dissertation proposes implementing and testing these existing RSU-based localization methods and security schemes. *The second one* (the RSU-enabled detection component) runs a real-time detector to monitor whether a received GPS measurement is spoofed or not. Based on the secure RSU data, the detection component computes multiple features, with which a real-time detector, based on the Isolation Forest, is constructed to detect GPS spoofing. Once spoofing is detected, GPS measurements are isolated, and the potentially compromised location estimator is corrected using the RSU data. This is *the third component* of the infrastructure-enabled defense method. Each of the three components is introduced in detail below.

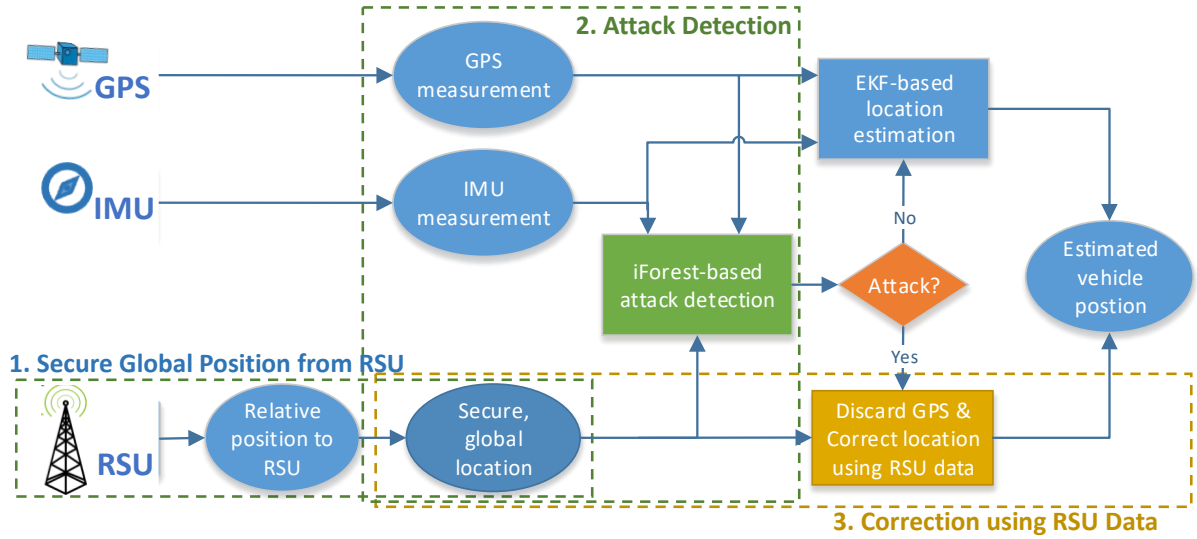

Figure 24. Infrastructure-enabled solution for GPS spoofing detection and correction.

1) *Secure RSU Data from the Infrastructure*

- *Design of secure RSU data*

Methods for obtaining secure RSU data include two major aspects: (i) what data to collect and how to collect them; and (ii) how to secure data collection and transmission. We focus on (i) in this study. For (ii), as discussed in Section 5.1.3, we apply the AES scheme, one of the state-of-

the-art encryption methods, to design dedicated secure channels for secure data collection and transmissions, focusing on testing its performance in spoofing detection and correction in Section 5.1.4.

The design of secure RSU data ensures that a vehicle can use RSU data to obtain a global position measurement similar to GPS, denoted as $\boldsymbol{x}_k^{RSU}$. This has been extensively studied in the study area of GPS-free localization (Adegoke et al., 2019; Khattab et al., 2015; Kuutti et al., 2018). A common practice is to first estimate the vehicle's relative position to the RSU via ranging methods and then compute the vehicle's global position given the (global) coordinates of the RSU (Hasan et al., 2020). In a ranging method, the distance between a radio transmitter (the RSU here) and a receiver can be inferred from the properties of the radio wave observed at the receiver (Khattab et al., 2015). Note that this distance is termed as *range* following the literature. The widely known ranging methods include those collecting and utilizing *received signal strength (RSS)*, arrival time or arrival angle (Adegoke et al., 2019; Alnasser et al., 2019). For CAVs that can communicate with RSUs, such range information can be readily available on the vehicle side. Following (Adegoke et al., 2019), we use $M(\bullet)$ to express a ranging method that obtains the range information z_k^{RSU} at time *k*:

$$\boldsymbol{z}_k^{RSU} = M(\boldsymbol{x}_k) + \boldsymbol{e}_k^{RSU}. \tag{58}$$

Here, $M(\bullet)$ is essentially a measurement model depending on the vehicle's (true) global position $\boldsymbol{x}_k$ and the RSU's coordinates Crd^{RSU}. e_k^{RSU} is the measurement noise in a Gaussian distribution with covariance matrix σ^{RSU}. In (Adegoke et al., 2019), a recent review of RSU-assisted localization methods is provided, which vary with the RSU data types and configurations of signal transmitters on RSUs and receivers on vehicles. There are also real-world implementations in GPS-absent environments (e.g., Waze's Beacon program to provide navigation for drivers

underground (Waze, 2021)). The RSU-assisted localization methods could reach an accuracy in centimeters, much higher than that of GPS (Khattab et al., 2015).

In this study, we implement an efficient and low-cost V2X-based vehicle localization method by Ma et al. (Ma et al., 2019). See a discussion of its efficiency in terms of computational latency below. It is low cost as it needs only a single data transmitter on the RSU side and a single receiver on the vehicle (i.e., it is similar to and can be implemented via the current V2X framework), compared with other ranging methods using multiple transmitters or receivers to collect information such as angle of arrivals (Ma et al., 2019). Ma et al. (Ma et al., 2019) assumes that the RSU broadcasts its coordinates, and a vehicle receives the message and extracts associated range information (i.e., the relative distance information) based on RSS only. Then the vehicle computes its global position $\boldsymbol{x}_k^{RSU}$ using a sequence of range information z_k^{RSU}. Therefore, this method may be readily deployed based on the current V2X systems without additional hardware requirements (the range information does need to be extracted from the receiver on the vehicle side). Omitting the details, we denote this method with function $G(\bullet)$ (Ma et al., 2019):

$$(\boldsymbol{x}_k^{RSU}, \boldsymbol{R}_k^{RSU}) = G([z_k^{RSU}, z_{k-1}^{RSU}, \dots, z_{k-o}^{RSU}], Crd^{RSU}, [\boldsymbol{u}_k, \boldsymbol{u}_{k-1}, \dots, \boldsymbol{u}_{k-o}]). \quad (59)$$

Note that $[\boldsymbol{z}_k^{RSU}, \boldsymbol{z}_{k-1}^{RSU}, \dots, \boldsymbol{z}_{k-o}^{RSU}]$ of length o represents the sequence of range information associated with the messages from an RSU. $[\boldsymbol{u}_k, \boldsymbol{u}_{k-1}, \dots, \boldsymbol{u}_{k-o}]$ represents a sequence of local measurements containing either speeds or local displacements. These local measurements can be easily accessible from either the vehicle's own wheel encoder or IMU. The covariance matrix $\boldsymbol{R}_k^{RSU}$ considers the uncertainty associated with the estimated position $\boldsymbol{x}_k^{RSU}$, which may be affected by the sequence length and noises in the range information. It is reported that the error of $\boldsymbol{x}_k^{RSU}$ is less than one meter. In our study, we conduct sensitivity analysis in Section 5.1.4 to test whether RSU-assisted location accuracy will play a role in detecting and correcting GPS spoofing attacks.

Lastly, the latency needs to be considered when implementing the AES scheme to set up the secure channel between an RSU and vehicles. Here latency stems from three sources: the communication latency, the latency due to encrypting and decrypting the transmitted data, and the computational time to derive the vehicle's global position. One main contribution to the communication latency is the V2X technology involved, such as the Dedicated Short Range Communication (DSRC) and the emerging 5G-based Cellular-V2X (C-V2X) system. Previous studies have reported that the DSRC communication latency ranges from 10ms to 100ms (Bey and Tewolde, 2019; Mannoni et al., 2019; Shimizu et al., 2019) and the C-V2X communication latency would not exceed 60ms even when there are 150 vehicles in the same communication channel (Bey and Tewolde, 2019; Kutila et al., 2019; Wang et al., 2019). In our implementation, the run times for encrypting/decrypting the transmitted data and deriving the vehicle's global position are negligible (0.60ms and 0.13ms, respectively), when evaluated from an average of 1000 runs on a personal computer (with a 3.60GHz AMD Ryzen 7 CPU). This suggests that the latency of the designed secure RSU data is dominated by communication latency. In this research, we set 100ms, the largest reported communication latency in the numerical experiments in Section 5.1.4.

- *RSU-based location prediction*

The relative vehicle position measured by RSU, $\boldsymbol{z}_k^{RSU}$, would not always be available, depending on the availability of RSUs along the road. Specifically, due to a possible limited budget in a real-world setting, RSUs are most likely spatially sparse in the road network and RSU data can only be available when vehicles are within an RSU's service range. An example is given in Figure 25, where four RSUs are installed along a road segment and RSU data are only available close to an RSU, depending on the V2X communication range. In this study, we assume the distance between two consecutive RSUs, denoted as D_{RSU}, is uniform, and the service range d_{RSU}

is fixed. In Section 5.1.4, we conduct sensitivity analyses on how the spacing of RSUs will impact the performance of the proposed methods.

If RSU data are not available or have a delay, we utilize the last available RSU data and vehicle motion model to predict a vehicle's location, which enables us to continuously monitor GPS measurements and timely detect attacks. The prediction should not involve GPS measurements that may have been compromised before being detected. However, since the vehicle location may change dramatically following commands from the vehicle's actuator (e.g., throttle, brake and steer), predicting the vehicle location can be challenging.

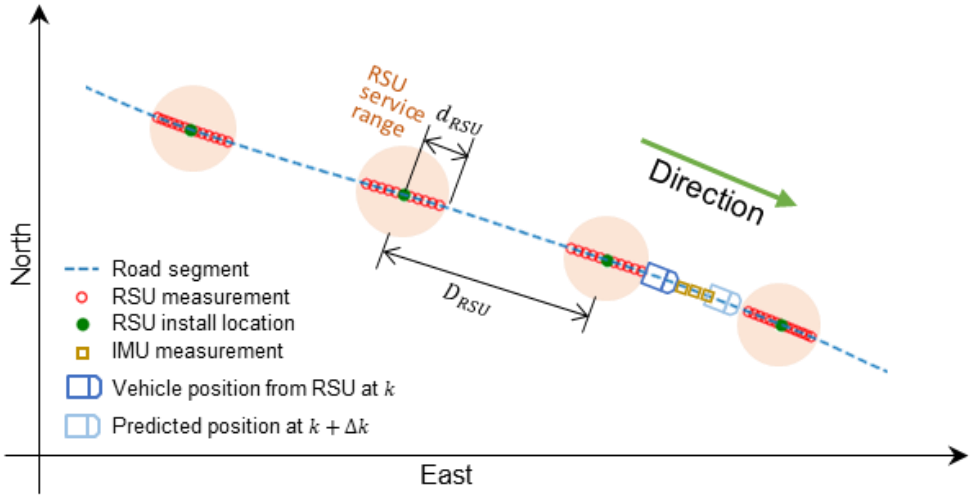

Figure 25. Illustration of RSU measurements along a road segment and RSU-based location prediction

We build an RSU-based prediction model leveraging RSU data and the vehicle motion model to address this challenge. Specifically, given the most recent vehicle (global) position information ($\boldsymbol{x}_k^{RSU}$; see Equation (8)) enabled by the RSU at time k, we predict vehicle location at $k + \Delta k$ (Figure 25). For this, we build and initialize a vehicle motion model (Equation (53)) with the most recent vehicle position estimated from RSU data $\boldsymbol{x}_k^{RSU}$ and then iterate it using IMU data $\boldsymbol{u}_k'$ ($t \in [k+1, k+\Delta k]$) as the input. Note that besides predicting the vehicle location, we also propagate

the errors in IMU data to gain the prediction uncertainty that is represented by a covariance matrix $\boldsymbol{P}_k^{RSU}$. The iterations of $\boldsymbol{x}_k^{RSU}$ and $\boldsymbol{P}_k^{RSU}$ are expressed in Equation (60). We will use this prediction model in the following sections to detect GPS spoofing and to correct the vehicle location when GPS spoofing is detected.

$$\begin{aligned} \boldsymbol{x}_t^{RSU} &= f(\boldsymbol{x}_{t-1}^{RSU}, \boldsymbol{u}_k'), \\ \boldsymbol{P}_t^{RSU} &= \boldsymbol{F}_{t-1}\boldsymbol{P}_{t-1}^{RSU}\boldsymbol{F}_{t-1}^T + \boldsymbol{L}_{t-1}\boldsymbol{Q}\boldsymbol{L}_{t-1}^T \end{aligned} \qquad t \in [k+1, k+\Delta k] \qquad (60)$$

Here, $\boldsymbol{F}_{t-1} = \frac{\partial f_{t-1}}{\partial \boldsymbol{x}_{t-1}}|_{\boldsymbol{x}_{t-1}^{RSU}}$ and $\boldsymbol{L}_{k-1} = \frac{\partial f_{k-1}}{\partial \boldsymbol{w}_{k-1}}|_{\boldsymbol{x}_{t-1}^{RSU}}$ are the partial derivative matrices corresponding to the state $\boldsymbol{x}$ and input noises $\boldsymbol{w}$ that are obtained by linearizing the motion model Equation (53) at the previous state. The iteration starts right after the initialization at k and yields at $k + \Delta k$. The algorithm for implementing RSU-based prediction is summarized in Figure 26.

Input: RSU and IMU measurements.
Output: The predicted vehicle location.

Process:
Step 1: Initialize the predictor using a RSU measurement at time step *k* following Eq. (8);
Step 2: **While** (*the trip continues*):
k = *k* + 1;
If (*IMU measurement is received*):
Update the prediction following Eq. (9);
If (*RSU measurement is received*):
Go to step 1;
If (*the request of RSU-based location prediction is received*):
Yield the predicted vehicle location;

Figure 26. Algorithm for generating RSU-based location prediction without involving GPS data.

2) iForest Model-based Attack Detection

Given the RSU data, we propose a detector that continuously monitors GPS measurements and detects GPS spoofing. The basic idea is that if the GPS measurement deviates too much from

the predicted vehicle location, the GPS will likely be spoofed. We formulate it as a real-time anomaly detection problem, containing two parts: 1) generating real-time features and 2) building a machine learning model that determines whether a GPS measurement is anomalous or not using features at *k*. Each part is designed as follows.

- *Feature generation*

The classical feature NEES:

We start with the classical feature for GPS spoofing detection, called NEES (see Chapter 2). It is computed as the normalized deviation of the received (possibly spoofed) GPS measurement $\tilde{\boldsymbol{z}}_k^{GPS}$ from the predicted location $\hat{\boldsymbol{x}}_k$, denoted as $\boldsymbol{r}_k^{GPS}$, as follows.

$$\begin{aligned} \boldsymbol{r}_k^{GPS} &= \tilde{\boldsymbol{z}}_k^{GPS} - \boldsymbol{H}\hat{\boldsymbol{x}}_k \\ \boldsymbol{S}_k^{GPS} &= \boldsymbol{H}\hat{\boldsymbol{P}}_k\boldsymbol{H}^T + \boldsymbol{R}^{GPS} \\ NEES_k^{GPS} &= (\boldsymbol{r}_k^{GPS})^T(\boldsymbol{S}_k^{GPS})^{-1}\boldsymbol{r}_k^{GPS} \end{aligned} \tag{61}$$

Note that $\boldsymbol{H}$ and $\hat{\boldsymbol{P}}_k$ are defined in the previous section, and $\boldsymbol{S}_k^{GPS}$ here is a covariance matrix reflecting the uncertainty associated with $\boldsymbol{r}_k^{GPS}$.

It has been proven that if the noises in measurements follow a normal distribution, NEES follows a χ^2 distribution (Chen et al., 2018). Therefore, in previous studies, the χ^2-test-based detection using $NEES_k^{GPS}$ is often applied to detect GPS spoofing. However, NEES could be impacted by noisy GPS measurements making it hard to differentiate attacks from noises (Liu et al., 2011). Furthermore, the χ^2-test-based detection could be ineffective in facing stealthy attacks (Shen et al., 2020b). This is because attackers could inject a sequence of false information into the authentic GPS measurements; each piece of false information alone may not lead to a large enough NEES to trigger the alarm, but these errors together could successfully deviate the vehicle. If this

happens, the χ^2-test-based detector itself may also be compromised, making it less effective to detect spoofing attacks. We demonstrate this phenomenon later in our experimental study.

Features generated from RSU data:

The independent, secure measurements from the infrastructure (i.e., RSUs) enable us to design a more robust detector. In particular, we can create new features based on the measurements from RSUs, without involving GPS measurements, to address the issues associated with NEES. A straightforward way to create new features is to compute the difference between RSU and GPS measurements. However, as noted earlier, measurements from RSUs and GPS may not be at the same frequency, with the former not always being available. As a result, the two would not be directly comparable.

We utilize the RSU-based location prediction discussed in the previous section to address this issue. The predicted location is generated whenever a GPS measurement is received and needs to be validated. Then, new features are created by comparing the GPS measurement with RSU-based prediction in Equation (60). Since the prediction in (8) does not involve GPS measurements, these features are 'protected' as they are immune to GPS spoofing attacks. Specifically, using the RSU-based location prediction $\boldsymbol{x}_k^{RSU}$ and the associated covariance matrix $\boldsymbol{P}_k^{RSU}$, we first compute the residual between the GPS measurement and the prediction $\boldsymbol{r}_k^{GPS}$ as well as the uncertainty of the residual $\boldsymbol{S}_k^{RSU}$, following Equation (62). Then we generate two new (scalar) features r_k^{RSU} and S_k^{RSU}, as shown in Equation (63).

$$\begin{aligned} \boldsymbol{r}_k^{RSU} &= \tilde{\boldsymbol{z}}_k^{GPS} - \boldsymbol{H}\boldsymbol{x}_k^{RSU} \\ \boldsymbol{S}_k^{RSU} &= \boldsymbol{H}\boldsymbol{P}_k^{RSU}\boldsymbol{H}^T + \boldsymbol{R}^{RSU} \end{aligned} \tag{62}$$

$$r_k^{RSU} = \left\| \boldsymbol{r}_k^{RSU} \right\| \tag{63}$$

$$S_k^{RSU} = \left|\boldsymbol{S}_k^{RSU}\right|$$

Here, $\|\bullet\|$ and $|\bullet|$ compute the L2 norm of a vector and the determinant of a matrix, respectively.

In the following, we build an unsupervised learning algorithm to detect GPS spoofing utilizing the computed features.

- *Building an Isolation Forest as the detector*

The attack detection is treated as a real-time anomaly detection problem, for which we apply an unsupervised machine learning model to learn anomalies from the data. Specifically, we detect GPS spoofing by building an Isolation Forest (iForest) that takes all the above features $\boldsymbol{A}_k = (NEES_k^{GPS}, r_k^{RSU}, S_k^{RSU})$ at time *k* as the input. iForest produces binary outputs, one indicating being under attack (denoted as $\delta_k = 1$) and the other being benign (denoted as $\delta_k = -1$). Compared with other unsupervised learning methods, iForest has multiple advantages (Liu et al., 2012). First, it has been shown superior performance in detecting anomalies in extensive empirical studies. Second, iForest is easy to train in terms of selecting hyperparameters and can scale up to massive applications due to its linear time complexity and low memory consumption, making it suitable to run on vehicles whose resources are often constrained.

The intuition behind iForest is that anomalous (or malicious) samples are easier to separate (i.e., isolate) from others compared with benign samples. In order to isolate a sample, the algorithm recursively generates partitions on all the samples by randomly setting a split (e.g., a threshold with a random feature) until all samples are separated. The recursive partitioning process is represented by growing a tree structure named *Isolation Tree* (iTree), with the leaves (or terminating nodes) being separated samples and intermediate nodes being attribute splits. Then, the length of the path to reach a sample starting from the root of an iTree approximates the number of partitions required to isolate the sample; a short length suggests a sample suspicious to be

anomalous. By constructing a large number of (random) iTrees based on the training dataset, we build an iForest. Using this iForest, we can identify samples that tend to have shorter path lengths in iTrees than others as anomalous. Anomaly detection with iForest consists of two stages: 1) a training dataset is used to build a forest of iTrees (i.e., iForest), and 2) each testing sample is passed through these iTrees, and an average anomaly score is assigned to the sample, which is further classified as a binary value. Readers are referred to Liu et al. (2012) for more details.

As an example, Figure 27A shows a set of four GPS measurements and their features, among which z3 (in red) is a spoofed one. Figure 27b shows the iForest where each node in an iTree is a random condition for partition. If a GPS measurement does not satisfy this condition, it falls into its left child node, otherwise in its right child node. For instance, given the condition on the root node of the first tree being $NEES_k^{GPS} < 5$, only z3 falls to the left child node since the feature $NEES_k^{GPS}$ associated with z3 does not satisfy this condition. Following the similar process, we obtain random trees to form an iForest. The path length of z3 is around 1 in each of the iTrees, shorter than the lengths of other GPS measurements, indicating z3 is spoofed.

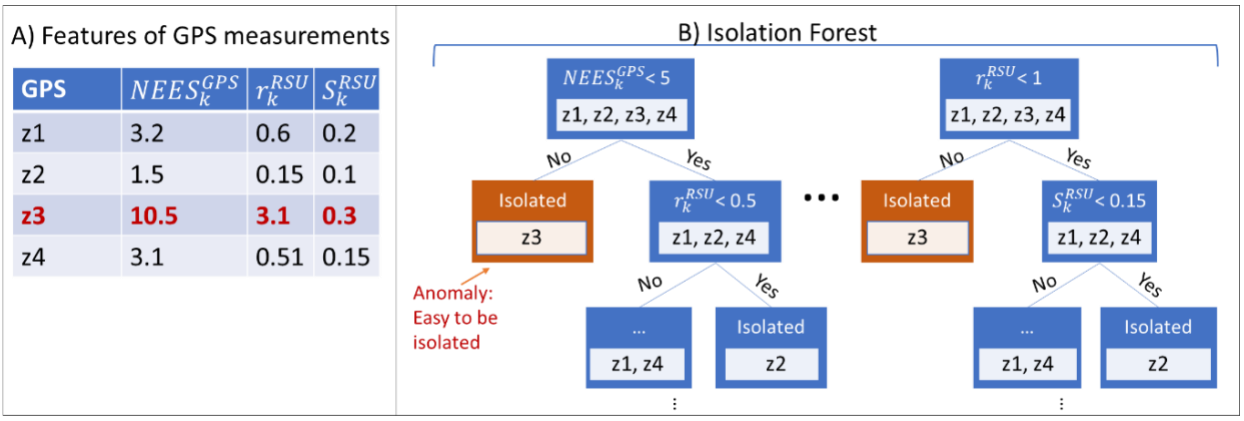

Figure 27. An illustrative example of how iForest detects spoofed GPS measurements.

Being an unsupervised learning method, the iForest can be trained without labeling the data; thus, the training data can be easily prepared. In this study, we generate training samples by

running vehicles and collecting the features at each time step. One advantage of iForest is that it works in scenarios where the training dataset contains no anomalies. Therefore, we could prepare training samples using historical data, which may or may not be attacked. Details of training iForest can be found in Section 5.1.3. The trained iForest can then be applied to detect GPS spoofing attacks in real-time. As expressed by Equation (64), to check whether the GPS measurement at time *k* is spoofed, we compute a set of real-time features $\boldsymbol{A}_k$ and input them to the trained iForest. An attack is detected if δ_k =1.

$$\delta_k = iForest(\boldsymbol{A}_k)\,,\ \ \delta_k \in \{-1, 1\}\,. \tag{64}$$

In applications where GPS noise is large, we improve the robustness of the iForest-based detector by accounting for the temporal pattern of the features (Ding and Fei, 2013). Specifically, we apply a sliding window to use not only the features at time *k* but also the ones at the previous time steps. In our experiment study where GPS noises are assumed large, features at the previous two steps (i.e., $\boldsymbol{A}_{k-2}, \boldsymbol{A}_{k-1}$) are incorporated to detect attacks at time *k*, as it is not common to observe three outliers consecutively. One may adopt a wider sliding window at the cost of a higher false negative rate; Figure 28 summarizes the algorithm for attack detection. The iForest model is implemented by *IsolationForest* in the Python package *sklearn* (Pedregosa et al., 2011).

Input: The trained detector $iForest$,
GPS measurement at time *k*,
Features computed at previous time steps.
Output: a binary value, i.e., 1 if GPS measurement at k is spoofed or -1 otherwise.

Process:

Step 1: Feature generation

a). Retrieve location estimation from EKF;

b). Compute $NEES_k^{GPS}$ following Eq. (10);

c). Retrieve RSU-based location prediction from algorithm 1;

d). Compute r_k^{RSU} and S_k^{RSU} following Eq. (11) and (12);

e). Obtain the features at time k: $\boldsymbol{A}_k = (NEES_k^{GPS}, r_k^{RSU}, S_k^{RSU})$;

Step 2: Determine whether the GPS measurement is spoofed or not

f). Retrieve features computed in the previous two time steps, i.e., $\boldsymbol{A}_{k-2}$ and $\boldsymbol{A}_{k-1}$;

g). Apply $iForest$ on the features ($\boldsymbol{A}_{k-2}$, $\boldsymbol{A}_{k-1}$, $\boldsymbol{A}_k$) and gain a binary output.

Figure 28. Algorithm for GPS spoofing detection.

3) *Infrastructure-enabled Correction*

The above detector continuously monitors the GPS measurements to identify potential spoofing attacks. If no attack is detected, the vehicle relies on EKF for location estimation, i.e., fusing IMU and GPS measurement to estimate location with high frequency. In the proposed method, measurements from RSUs will also be used to correct vehicle positions; the correction is triggered either a) when RSU data is received; or b) when GPS spoofing starts as indicated by the detector. In the second case, the RSU-based location predictions will be used for correction if no RSU measurement is available (e.g., an attack is detected when a vehicle is outside of the service range of RSUs). We introduce each case in detail in the following.

- *When RSU data is received*

When a vehicle enters the service range of an RSU, the vehicle periodically obtains measurements from the RSU, which can be used to correct the location estimation. The correction is done by directly initializing the state of EKF ($\widehat{\boldsymbol{x}}_k, \widehat{\boldsymbol{P}}_k$) following Equation (65).

$$\begin{aligned} \widehat{\boldsymbol{x}}_k &= \boldsymbol{x}_k^{RSU} \\ \widehat{\boldsymbol{P}}_k &= \boldsymbol{P}_k^{RSU} \end{aligned} \tag{65}$$

Here, ($\boldsymbol{x}_k^{RSU}, \boldsymbol{P}_k^{RSU}$) is the secure location estimation using RSU data following Equation (60). Note that an alternative way to correct vehicle position using RSU data is to follow the EKF's update step introduced in the following section, i.e., computing a Kalman gain and updating the EKF state leveraging the difference between the RSU measurement and the predicted location. However, this alternative may not be reliable in stealthy attacks, which may bypass the attack detector and gradually corrupt the EKF (Shen et al., 2020b). As a result, following the update step of a corrupted EKF would be inefficient. Therefore, the proposed method via direct initialization can effectively remove the negative effects of attacks.

- *When GPS spoofing is detected*

When the detector detects that an attack starts, besides isolating the GPS sensor, we correct the EKF estimator as well. If RSU data is available at this moment, Equation (65) will be followed to correct the EKF location estimator; if not, the predicted location yielded from the RSU-based prediction model will be used. Specifically, when GPS spoofing is detected starting at $k + \Delta k$ but RSU data is not available, the predicted position $\boldsymbol{x}_{k+\Delta k}^{RSU}$ and its covariance matrix $\boldsymbol{P}_{k+\Delta k}^{RSU}$ following Equation (65) are used for correcting the EKF. Similarly, the correction is done by directly initializing the EKF state.

Note that since the RSU-based prediction model does not involve GPS measurements that may have been falsified, the predicted location is able to correct errors resulting from a delayed

detection, where the EKF estimator may have been compromised already. We show in our experiments that this brings benefits in defending stealthy attacks where the attacks are often detected with a lag. This is distinctively different from existing spoofing defense methods without RSU data: unable to remove the negative effect of the spoofed GPS measurements, the vehicle locations produced by these existing methods would still remain deviated (from the true locations) even the vehicle successfully detects the attacks and discards incoming GPS measurements at a later time.

5.1.4. Results from Defending against GPS Spoofing

1) Experiment Settings

General settings

We test the infrastructure-enabled solution method for GPS spoofing detection and correction using both simulation and real-world data. The simulation model generates various driving scenarios and collects vehicle trajectories using Simulation of Urban Mobility (SUMO), while the real-world data consist of GPS trajectories from GPS devices installed on both trucks and passenger vehicles. Taking a trajectory as input, the MATLAB Navigation Toolbox (MNT) is used to simulate necessary sensor measurements along the trajectory, including local (e.g., IMU) and global measurements (e.g., RSU data). The GPS measurements are manipulated following the attack models (Section 5.1.2) to simulate GPS spoofing attacks.

The SUMO model is built for the multi-mode traffic in the Downtown Seattle area (Figure 29A). Fifty-three passenger vehicles are randomly selected for testing. These simulated trajectories allow us to capture diverse driving scenarios, including highway and local street driving, where both road geometries and vehicle dynamics vary considerably. As we show later, the diversified driving scenarios enable us to conduct sensitivity analysis on factors that may affect the

performance of the proposed method. On the other hand, real-world GPS data contains trajectories from 15 vehicles, which are less diverse but more likely to reflect real-world driving scenarios.

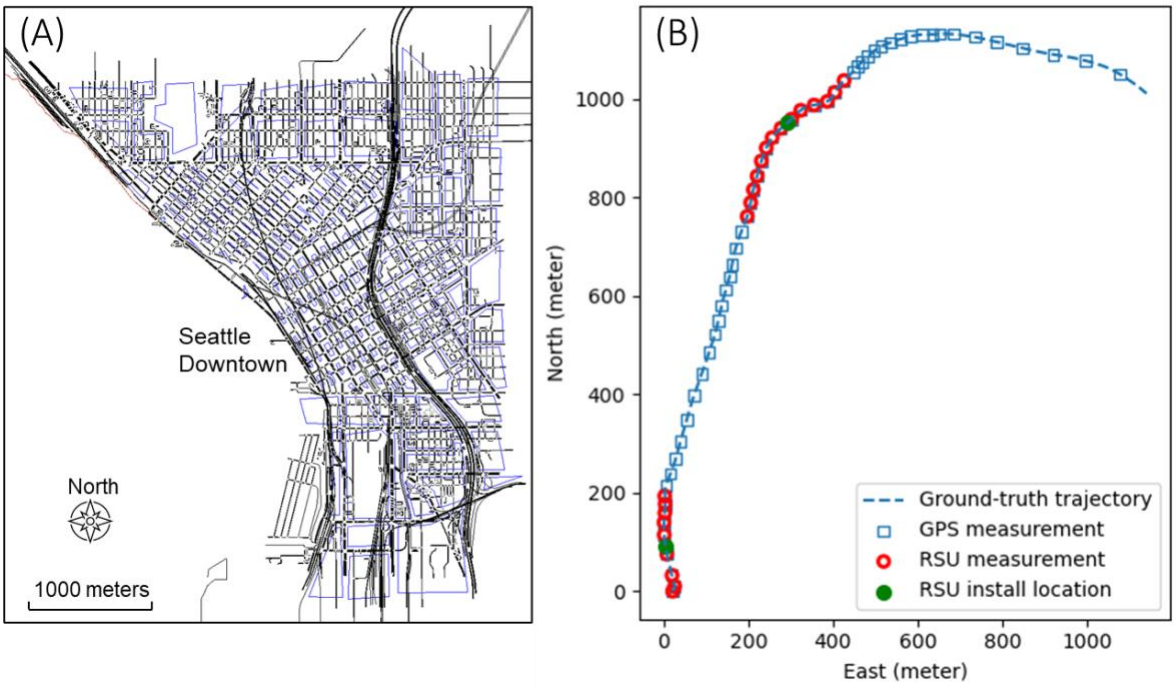

Figure 29. (A) Road network of Downtown Seattle in SUMO simulation; (B) Illustration of sensor measurements along a simulated trajectory.

MNT provides a rich set of tools and algorithms for vehicle navigation applications. The parameters of the IMU and GPS sensors (e.g., accuracy levels and resolutions) are set as MNT's default values, which reflect real-world sensor properties to a large extent. See details of MNT's sensor models in MATLAB documentation (MATLAB, 2021). IMU and GPS measurements are sampled at 10Hz and 1Hz, respectively. RSUs are located along the road at an equal distance (1500 meters), and the service range of an RSU is represented by a circle with a radius of 500 meters centering at the RSU. Under the service range of an RSU, radio signal-to-noise ratio (SNR) in dB is simulated using the ground-truth range (i.e., the distance between the vehicle and the RSU) and following the measurement model $SNR = 10\log_{10}(|\boldsymbol{z}^{RSU}|^2/(\sigma^{RSU})^2)$ as in (Ma et al., 2019) (essentially the reverse of the ranging method). Here, $|\boldsymbol{z}^{RSU}|$ is the Euclidean distance between the

vehicle and RSU and σ^{RSU} represents the uncertainty (see Section 5.1.3). The encryption, decryption and transmission process for data security is simulated via an AES scheme assuming 100ms latency as noted earlier. Based on computations in Section 5.1.3, a secure RSU-based vehicle location is generated when a GPS measurement is received.

Figure 29B illustrates an example of a ground-truth trajectory together with IMU, GPS and RSU measurements. The measurements are visualized at where they are received. It can be observed that GPS measurements are periodically received along the trajectory while RSU measurements are not spatially continuous but clustered around where RSUs are installed.

To simulate GPS spoofing, we randomly select the start time and duration of the attack (following a uniform distribution ranging from 5 to 35 seconds). The attack modifies the true GPS measurements and passes the modified measurements to the vehicle for location estimation. We simulate the two types of attacks as discussed in Section 5.1.2. For the constant bias attack, we modify GPS measurements to deviate them by four meters, which is roughly the lane width. For the stealthy attack, *m* and *n* are set at 1.0 and 1.07, respectively. We choose the two values so that the maximum deviation is comparable with the one in the constant bias attack (e.g., four meters) for an average attack duration of 20 seconds. These values also approximate the ones used in the original study by Wang et al. (2021b).

Experiment design and metrics

Since spoofed GPS measurements and the true trajectory are known in our testing, we can evaluate the performance of the proposed methods by checking whether the spoofed GPS measurements can be detected and computing the location estimation error. Here we demonstrate the effectiveness of the methods in both attack scenarios. In each scenario, we test the methods for detecting attacks and correcting vehicle locations. The evaluation is done by averaging the

performance of all the trajectories with random attack times (i.e., random attack starting time and attack duration). The average performance is then compared with benchmark methods, including the conventional χ^2-test-based detector and the state-of-the-art CUSUM detector for GPS spoofing detection (Section 2.2). As discussed earlier, the CUSUM detector requires tuning parameters. We do this by searching around the parameters suggested in the original work (Wang et al., 2021b) and taking the ones that yield the best performance in our experiments. For the iForest model, the training data is collected by running vehicles without GPS spoofing, and its hyperparameters follow the default settings in the *sklearn* implementation.

We then conduct sensitivity analyses by varying the distance between RSUs, exploring the hypermeters of iForest, and investigating the performance in various driving scenarios. The sensitivity analyses enhance our understanding of the properties of the proposed infrastructure-enabled methods and provide insights on deploying RSUs in real-world implementations.

Five performance metrics are adopted to evaluate the performance on each vehicle trajectory, including the *F1* score, *precision*, *recall*, *detection lag*, and *Rooted Mean Square Error* (*RMSE*) of the estimated locations. The first three are common metrics to measure the detection accuracy, ranging from 0 to 1. *precision* calculates the ratio of true positives over all the identified positives, and *recall* is the ratio of true positives to all ground-truth positives. A larger precision and a larger recall mean a lower false-positive (FP) rate and a higher true-positive (TP) rate, respectively. F1 score measures the balance between the false positives and true positives following Equation (66) (Bishop, 2006).

$$F1 = \frac{2 * precision * recall}{precision + recall} \tag{66}$$

A larger *F1* score means better detection performance in balancing false positives and true positives. The *detection lag* measures the time difference between the start time of an attack and the time when a detector first detects the attack.

RMSE measures the location estimation error along a trip by computing the distance between the estimated locations $\widehat{\boldsymbol{x}}_k$ and true locations $\boldsymbol{x}_k$:

$$RMSE = \sqrt{\frac{1}{K}\sum_{k=0}^{K} \|\widehat{\boldsymbol{x}}_k - \boldsymbol{x}_k\|_2} \tag{67}$$

Here, $\|\widehat{\boldsymbol{x}}_k - \boldsymbol{x}_k\|_2$ computes the distance between the estimated location and the true location at time *k*. *K* represents the duration of the trajectory.

2) Testing Results using Simulated Trajectories

Constant bias attacks

We first evaluate the proposed methods and benchmark methods under constant bias attacks. The mean detection accuracy and location estimation error are reported in Table 1. It can be found that all three methods can detect the start of attacks without any delay. Yet, given its strength in balancing false positives and false negatives, the infrastructure-enabled method performs the best among the three methods, while the CUSUM detection method outperforms the χ^2-test-based detector. Specifically, the infrastructure-enabled method gives an F1 score of 0.86, which is larger than 0.56 and 0.69 by the χ^2-test-based detector and CUSUM, respectively. This finding suggests that the proposed method can effectively detect the attacks, even though the measurements from RSUs are not always available.

The precision and recall of the proposed method indicate that the methods could nearly identify all the spoofed GPS measurements, while generating some false positives, which could be due to the noises in GPS sensors. As shown later in the sensitivity analysis, we can reduce the false

negatives while curbing false positives by tuning the hyperparameter of iForest. On the other hand, the low precisions by the χ^2-test-based detector and CUSUM suggest that they produce many false positives. With effective detection and correction, the infrastructure-enabled method can also dramatically reduce location errors compared to the benchmark methods, as shown in the table.

Table 1. Performance of the proposed and benchmark methods under constant bias attack

	χ^2-test-based	CUSUM	Infrastructure-enabled
F1 score	0.56	0.69	**0.86**
Precision	0.52	0.60	**0.77**
Recall	0.69	0.86	**0.99**
Detection lag	0	0	**0**
RMSE	5.74	4.53	**0.43**

- *An example trajectory under constant bias attack*

To help understand the differences between the three methods, we illustrate their performances on one randomly selected trajectory, as shown in Figure 30. Figure 30A compares the detection accuracies of the three methods. All of them can identify the attacks with no delay, while the CUSUM and χ^2-test-based detectors produce false negatives during the attack period. As a result, the proposed method yields an F1 score of 0.86, outperforming the CUSUM and χ^2-test-based detector with F1 scores of 0.82 and 0.57, respectively. χ^2-test-based detector produces multiple false positives when there is no attack. Being the second-best detector, CUSUM curbs false positives but misses some spoofed GPS measurements.

Figure 30B shows location estimation errors. The location errors resulting from the benchmark methods are much larger than those from the proposed method. The difference is due to two reasons. First, both the CUSUM and the χ^2-test-based detector generate false negatives and thus the location estimators are compromised by some of the spoofed GPS measurements (as the

detectors treat them as not spoofed). In contrast, the proposed method can accurately detect and discard spoofed GPS measurements and then relies on dead reckoning instead of following the spoofed GPS data for vehicle locations. Second, even when successfully detecting most spoofed GPS measurements, the benchmark methods do not correct errors accumulated in the compromised location estimators before spoofing was detected. As a result, the errors persist even after the spoofing attacks are detected and even after the attacks are completed. On the contrary, when the infrastructure-enabled method detects the attack, the vehicle location is immediately corrected by the RSU data, removing the impacts of the spoofed GPS.

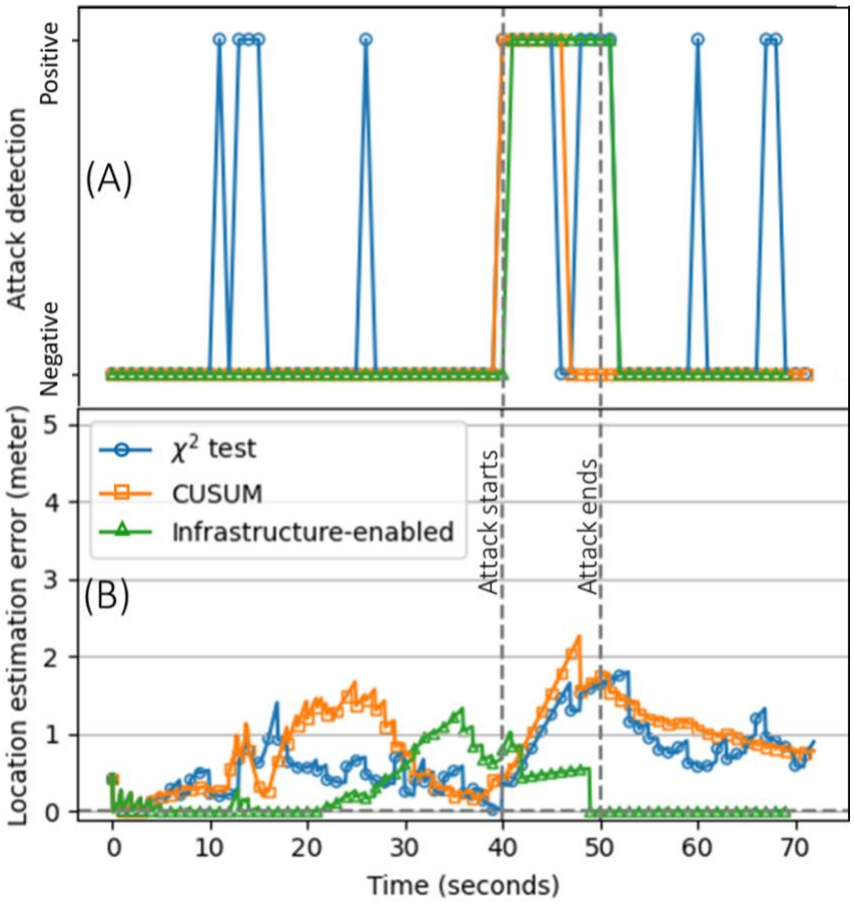

Figure 30. Performance under constant bias attack: an example trajectory. (A) Detection accuracy. Positive means a detector reports an attack; the legend is shown in the right figure. (B) Location estimation errors.

Stealthy attacks

Similarly, we evaluate the proposed and benchmark methods under stealthy attacks. The results are presented in Table 2. Again, the infrastructure-enabled method performs the best among the three methods under stealthy attacks. It gives an F1 score of 0.78, which is larger than 0.47 and 0.21 generated by the χ^2-test-based detector and CUSUM, respectively.

Table 2. Performance of the proposed and alternative methods under stealthy attack

	χ^2-test-based	CUSUM	Infrastructure-enabled
F1 score	0.47	0.21	**0.78**
Precision	0.53	0.30	**0.76**
Recall	0.49	0.22	**0.84**
Detection lag	3	14	**2**
RMSE (meter)	5.69	4.58	**0.42**

Compared with the performance under the constant bias attacks (Table 1), Table 2 provides several interesting findings. First, the performances of all three methods are worse than those under the constant bias attacks. This is reasonable as the stealthy attacks are more deceptive than the constant bias attacks. Noteworthy is that though being downgraded, the performance of the infrastructure-enabled method is still promising: the recall of 0.84 suggests that 84% of spoofed GPS measurements can be successfully detected. Second, it is interesting to notice that the CUSUM detector performs even worse than the χ^2-test-based detector in detecting stealthy GPS spoofing attacks, in contrast to the finding under the constant bias attacks. This is because of the underlying design mechanism of CUSUM: it depends on all the observations prior to the decision time for reducing false positives. Under stealthy attacks, this design has the side effect of increasing false positives, especially at the beginning of the attack when the deviation is small, as indicated by the long detection lag (Table 2). As a result, the EKF estimator could be corrupted

substantially before the attack is detected, leading to significantly compromised performance. We demonstrate this point in the following example trajectory.

An example trajectory under stealthy attacks We compare the performances of the three methods under stealthy attacks using a randomly selected trajectory (Figure 31). In this example trajectory, the proposed method yields an F1 score of 0.97, outperforming both the CUSUM (an F1 score of 0.37) and χ^2-test-based detector (an F1 scores of 0.58). It can be observed that CUSUM shows a long detection delay (about 20 seconds in this example), consistent with our observation earlier that CUSUM is not effective in detecting stealthy attacks that start with minor deviations. Figure 31B shows that the EKF location estimators in both the χ^2-test-based detector and CUSUM are corrupted, since the two detectors either fail to detect the attack timely or generate some false negatives. Furthermore, errors of the corrupted EKF location estimators in the benchmark methods continue to persist even when the attack is detected at a later stage. The corrupted location estimators can also compromise the attack detectors, which rely on the estimator outputs to detect spoofing attacks, further exacerbating the degraded performances of the two benchmark methods. Figure 31B shows that, even after the spoofing attack is ended, the χ^2-test-based detector and CUSUM continue to "claim" that GPS spoofing still exists (i.e., false positives) for about 10 seconds and 15 seconds, respectively. These false positives can, in turn, worsen the location estimation, as shown in the figure. In contrast, the infrastructure-enabled method results in significantly smaller location errors, because 1) spoofed GPS measurements are timely detected and isolated, and 2) corrections are made using data from RSUs, thus eliminating the potential impact of the compromised GPS data. These observations suggest that it is necessary to correct location errors after attacks are detected; the conventional treatment (i.e., isolating compromised data sources alone) is insufficient in doing so.

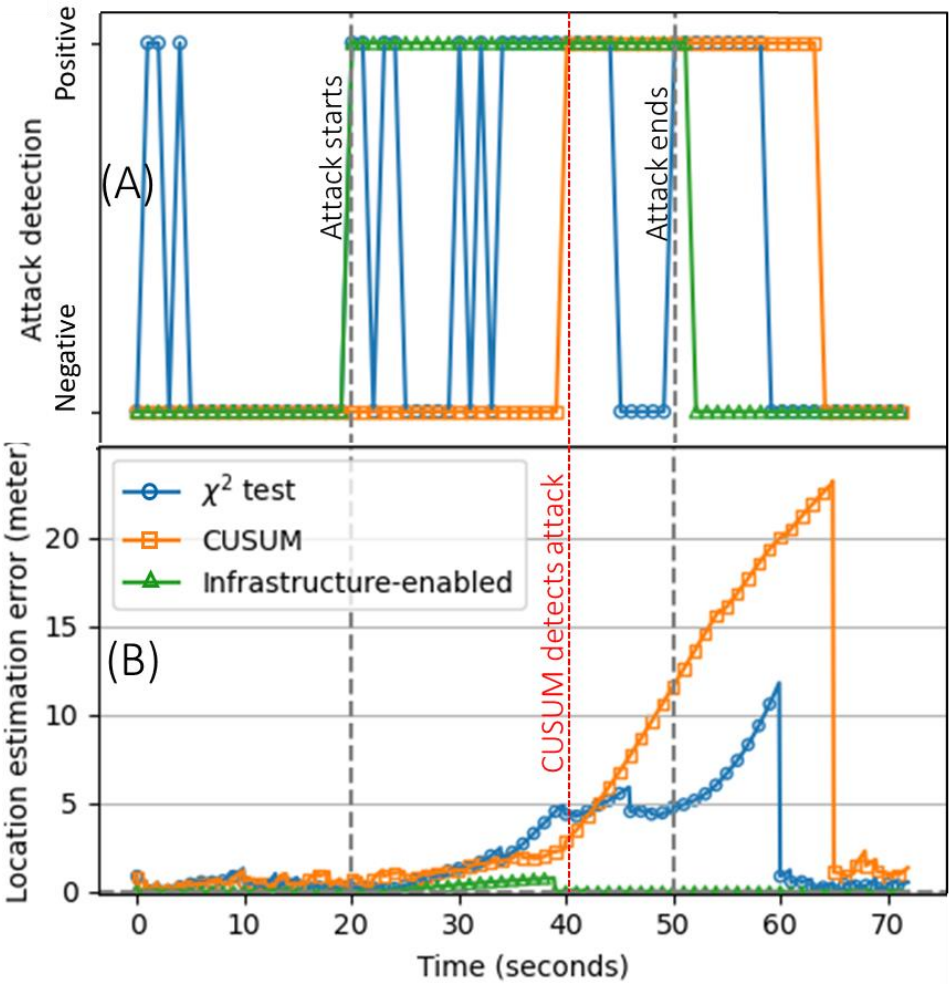

Figure 31. Performance under stealthy attack: an example trajectory. (A) Detection accuracy. Positive means a detector reports an attack. (B) Location estimation errors.

3) Testing Results using Real-world Trajectories

We further evaluate the proposed methods using real-world GPS trajectories under both constant bias and stealthy attacks. Similarly, the mean detection accuracy and location estimation error are reported in Table 3. It can be found that the infrastructure-enabled method outperforms the benchmark methods under both types of attacks. The F1 scores are close to those of the tests using simulated trajectories, suggesting the proposed method is also effective in dealing with real-world trajectories. Specifically, under constant bias attacks, the infrastructure-enabled method gives an F1 score of 0.93, which is significantly larger than the 0.54 and 0.66 resulting from the

χ^2-test-based and CUSUM detectors, respectively. In particular, the recall of the infrastructure-enabled method indicates that it can detect all spoofed GPS measurements with constant biases.

Table 3. Performances of the proposed and benchmark methods on real-world trajectories

		χ^2-test-based detector	CUSUM	Infrastructure-enabled
Constant bias attack	*F1 Score*	0.54	0.66	**0.93**
	Precision	0.64	0.61	**0.88**
	Recall	0.52	0.75	**1.00**
	Detection lag	0	0	**0**
	RMSE	2.52	1.90	**0.17**
Stealthy attack	*F1 score*	0.44	0.22	**0.73**
	Precision	0.64	0.54	**0.83**
	Recall	0.38	0.15	**0.67**
	Detection lag	6	16	**6**
	RMSE	2.02	1.19	**0.17**

The reduced F1 scores under stealthy attacks suggest that all the methods are less effective in detecting such attacks than detecting constant bias attacks. Yet, the proposed method still performs the best, with a mean F1 score of 0.73. Again, the CUSUM detector performs worse than the χ^2-test-based detector. We notice that the recall of the proposed method reduces to 0.67 in detecting stealthy attacks on real-world trajectories, which is significantly smaller than the recall of 0.84 for simulated trajectories. This suggests that the proposed method produces more false negatives when dealing with real-world trajectories. This is probably because the GPS measurements in real-world trajectories are noisier than simulated trajectories, resulting in an iForest detector that is less sensitive to large deviations from either the noises or the attacks. This effect also leads to longer lags of detecting attacks in dealing with real-world trajectories. In the following sensitivity

analyses, we show that these false negatives can be mitigated by adjusting the hyperparameters of the iForest detector. We omit the detailed presentation and discussion on individual vehicle trajectories as they are similar to those of the simulated trajectories (in Figure 30 and Figure 31).

4) Sensitivity Analysis

Multiple factors are related to the implementation of the infrastructure-enabled method, such as the distance between two consecutive RSUs and the hyperparameters of the iForest-based detector. These factors could directly affect the performance of the proposed method. Meanwhile, different driving conditions with various road geometries and mobility-related features (e.g., speed) may also play a role in influencing performance. We conduct sensitivity analyses of the infrastructure-enabled method in this subsection on these factors. Simulated trajectories are used for the analyses unless indicated otherwise.

Distance between two consecutive RSUs

Given its reliance on RSU data, the infrastructure-enabled method is expected to be influenced by the strategy of RSU deployment. Specifically, out of the service range of any RSUs, the vehicle relies on the predicted location using the RSU data for attack detection and correction. Table 4 shows the performance with various D_{RSU} under the constant bias attack and the stealthy attack. Note that we stop at 3000m as most of the trajectories (95% of simulated and 80% of real-world trajectories) are shorter than 3000m and a larger D_{RSU} does not reduce the performance further. For the same reason, both the simulation data and real-world data are used for the analysis: the results for D_{RSU}=1000m, 1500m, 2000m are generated using the simulation trajectories (which can simulate more varied traffic/driving conditions), and the results for D_{RSU}=3000m are generated using the real-world data (which has a larger percentage of trajectories longer than 3000m). As

expected, the performance downgrades as D_{RSU} increases. Yet, we observe that the infrastructure-enabled method still maintains an advantage over the benchmark methods as D_{RSU} increases.

Table 4. Influence of RSU spacing on the infrastructure-enabled method

		D_{RSU}=1000m	D_{RSU}=1500m	D_{RSU}=2000m	D_{RSU}=3000m
Constant bias attack	*F1 Score*	0.92	0.86	0.82	0.87
	Precision	0.86	0.77	0.72	0.80
	Recall	0.99	0.99	0.99	1.0
	Detection lag	0	0	0	0
	RMSE	0.10	0.43	0.54	0.43
Stealthy attack	*F1 score*	0.83	0.78	0.62	0.69
	Precision	0.83	0.76	0.64	0.75
	Recall	0.83	0.84	0.65	0.68
	Detection lag	3	2	6	6
	RMSE	0.10	0.42	0.60	0.43

We also investigate how a longer distance between RSUs leads to lower detection accuracy. Figure 32 presents the relationship between the F1 score and the time elapsed when an attack starts since the last visited RSU. It shows a trend that the F1 score reduces as a vehicle travels away from an RSU. This is because when the vehicle exits the RSU service range, the IMU measurements will be used for location prediction (Equation (60)), leading to increased location errors. As a result, the prediction uncertainty is larger when the vehicle is farther away from the last visited RSU, which affects the accuracy of attack detection that relies on the RSU-based prediction to compute features. This finding suggests that in scenarios where the RSU spacing is large, one could maintain a high detection accuracy by incorporating additional data sources to reduce the uncertainty of the predicted locations.

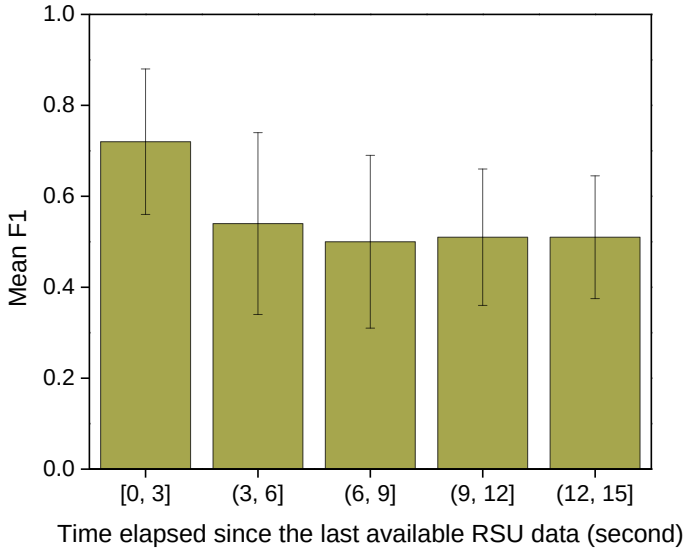

Figure 32**.** Relationship between F1 score and the time elapsed when an attack starts since the last visited RSU.

Hyperparameter of the attack detector

In the proposed infrastructure-enabled solution method, iForest serves as the core algorithm for attack detection. One key hyperparameter of iForest is *contamination* (denoted as α) that specifies the proportion of spoofed samples in the data set. The experiments above take the default setting 'auto' in *sklearn*, allowing it to determine α automatically based on the training data. We have observed promising results even with this default setting. Next, we investigate the sensitivity of the infrastructure-enabled method with α.

Table 5 summarizes the performance of the proposed method with different α's. Following *sklearn* implementation, α ranges from 0 to 0.5; 0 means no anomalies and 0.5 means that half of the data samples are anomalies. Therefore, a range from 0.1 to 0.3 in Table 5 captures a fairly large range of α. We can find that under all the settings, the infrastructure-enabled method outperforms the χ^2-test-based detector and CUSUM. Recalls remain unchanged in the constant bias attacks, suggesting that the proposed methods can robustly detect spoofed GPS measurements under this

type of attack for a wide range of α. Yet, under both constant bias and stealthy attacks, a larger α leads to more false positives, as indicated by the decrease in precision. On the other hand, a larger α brings benefits to detecting stealthy attacks, since 1) more spoofed GPS measurements can be detected (as indicated by the larger recall), and 2) the detection lag is shorter.

In summary, a proper α leads to a balance between false positives and false negatives. The proper value of α depends on the types of attacks: a small α is good enough for detecting constant bias attacks but encourages false negatives in stealthy attacks. Given the high threat of stealthy attacks, it would be beneficial to set a relatively large α to effectively detect this type of attack. In our experiments, a balance between false positives and negatives under the stealthy attack can be reached around α=0.2.

Table 5. Impacts of iForest hyperparameter on the proposed method

		α = 0.1	α = 0.2	α = 0.3
Constant bias attack	*F1 Score*	0.91	0.86	0.8
	Precision	0.86	0.77	0.69
	Recall	0.99	0.99	0.99
	Detection lag	0	0	0
	RMSE	0.25	0.43	0.31
Stealthy attack	*F1 score*	0.71	0.78	0.71
	Precision	0.8	0.76	0.62
	Recall	0.65	0.84	0.86
	Detection lag	6	3	2
	RMSE	0.35	0.42	0.45

Driving scenarios

Lastly, we investigate driving scenario-related factors that may affect the performance of the infrastructure-enabled method. Figure 33 compares the detection accuracy in the F1 score when

vehicles drive on highways and local streets. The results show that the detection accuracy on highways is slightly lower than that on local streets. This is likely due to the challenge in estimating vehicle locations based on dead-reckoning in high-speed driving on highways, leading to unreliable RSU-based locational predictions and thus worse detection accuracy. Besides the travel speed, the road curvature of segments along the vehicle's trajectory and the steering rate as the vehicle travels may also make a difference in detecting attacks on highways and local streets. These observations suggest that an adaptive strategy (e.g., based on the road type and geometry, and traffic speed, etc.) to deploy RSUs would be better than placing RSUs at fixed (and constant) spacings.

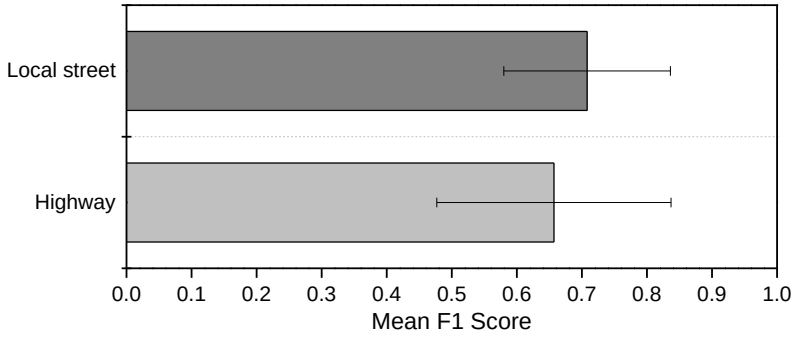

Figure 33**.** Comparison of detection accuracies on local streets and highways

5.1.5. Summary

This section proposes an infrastructure-enabled defense method that utilizes secure RSU data for detecting GPS spoofing and correcting location errors resulting from the spoofing. Timely detection is achieved by designing and training an iForest model using real-time features computed from both RSU data and (possibly spoofed) GPS data. Once spoofing is detected, GPS data is isolated and the compromised vehicle locations are corrected using the RSU data. Experimental results using both simulation data and real-world GPS data demonstrated that the proposed method enhances timely detection and correction even when RSU data is not spatially continuous. We

showed that IED is effective under state-of-the-art GPS spoofing attacks that were designed to be stealthy. Furthermore, sensitivity analyses produced insights into the RSU deployment strategies as well as into the impacts of iForest's hyperparameters and the accuracy of RSU-assisted localization on the IED's performance. By comparing with two benchmark methods, we showed that the IED method for GPS spoofing distinguishes itself from existing methods in two major aspects. First, it relaxes the requirement of vehicular sensors, making the defense more robust when dealing with spoofing attacks. Second, enabled by the secure RSU data, a simple detector based on an unsupervised learning algorithm (i.e., iForest) can effectively deal with GPS spoofing attacks.

5.2. Infrastructure-Enabled Defense against Attacks on Queue Length Estimation Model

5.2.1. Problem Statement

It has long been recognized that vehicular queue length is crucial to either signal performance measures in terms of vehicle delay and stops, or signal optimization. Therefore, protecting the queue length estimation model from attacks is important for traffic state estimation and traffic operation. We have briefly introduced the queue length estimation model in the previous section. It has been noted earlier that some vehicles' travel time information at the upstream (VTL1) and downstream (VTL2) are sampled and wirelessly collected to estimate queue lengths. The previous results have shown that data poisoning attacks can compromise the estimation model. Without detecting the attacks, the compromised queue length estimation could further mislead the adaptive signal controllers, possibly reducing traffic performance.

Our goal is to detect data poisoning attacks using infrastructure status data and existing or newly deployed secure data. Focusing on queue length estimation, we use the signal timing information and the travel times of (some) vehicles traversing the intersection collected from wired loop detectors to build a benchmark model (see details in the following section). These data enable us to estimate benchmark queue length using the knowledge of how queues accumulate and dissipate given signal timing information. In general, loop detector data and signal status data are collected via a wired system, which is relatively secure from data attacks. Based on this fact, we assume that the queue length estimation using the loop detector data and signal status data is safe from data poisoning attacks and can be used as the benchmark for attack detection. The resultant benchmark queue length, similar to the “ground truth” vehicle locations from RSU for GPS spoofing, can provide an effective means to detect data poisoning attacks. Learning models (e.g.,

isolation forest or DL models) may use this benchmark data and potentially compromised queue length to identify data attacks.

5.2.2. Preliminaries

1) Loop detector data and signal status data

The inductive loop detector has long been used as a standard data source for traffic surveillance. Such a device can provide several essential traffic status information, including occupancy, vehicle gap, flow and speed, etc. Many methods have been developed to estimate queue lengths using loop detector data; they are typically based on features of the evolution of traffic flows. At an intersection, traffic flows are interrupted by the queue accumulation and discharge process. The changes in traffic flows can be captured by the loop detectors. In general, as the queue forms and traffic slow down, the occupancy time (computed as the time difference of a vehicle's entry and exit time) at one detector would increase (see Figure 38 in the following section). If a vehicle in the queue sits still on the detector, the occupancy time often becomes significantly longer; so does the time gap between two consecutive vehicles. Together with the signaling time information (i.e., the starting and ending time of each signal cycle including each phase for each approach), queue lengths at each approach can be estimated, which will be introduced in the following section.

5.2.3. Methodology

1) Overview

The overall workflow of the proposed infrastructure-enabled solution to defend against attacks on queue length estimation is illustrated in Figure 34. It includes three components, each of which is introduced in detail below.

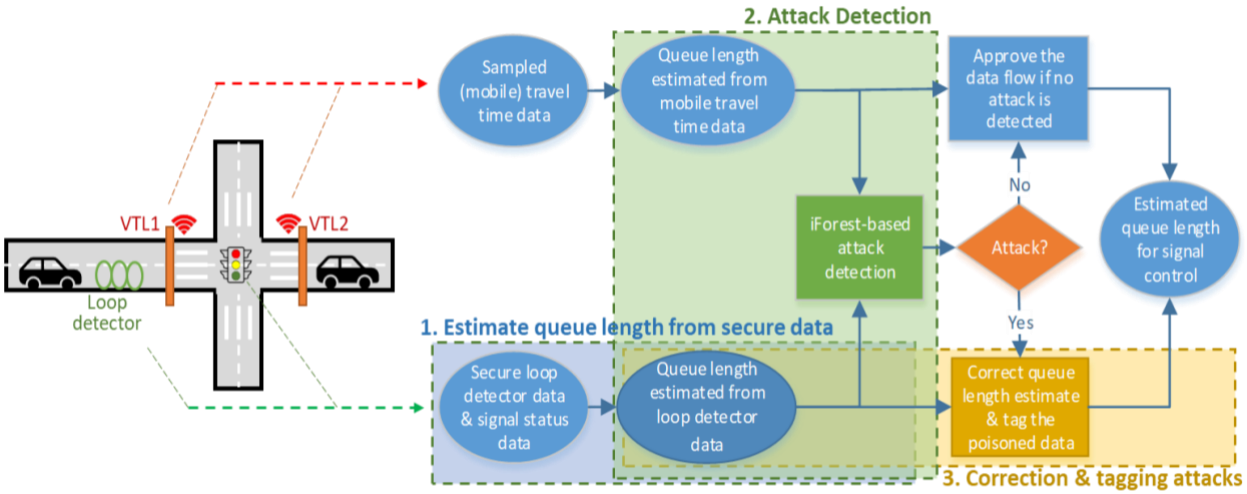

Figure 34. Illustration of the infrastructure-enabled defense against attacks on queue length estimation.

2) Estimating queue length using loop detector data and signal status data

Among the many methods of estimating queue lengths using loop detector data, this research refers to the method proposed by (Liu et al., 2009). The chosen method is more generalizable than others as it can estimate long queues (i.e., queues longer than the distance between the intersection stop line and loop detectors' position).

The basic idea is to utilize the signal status data (the event history of a traffic signal) to analyze the relationship between signal phase changes and traffic flow during the queue forming and discharging processes. Then, it focuses on identifying critical changes in traffic flow patterns characterized by some "breakpoints." These breakpoints have important physical meanings that connect to several types of traffic shockwaves. Together with the shockwave theory (Lighthill and Whitham 1955 and Richards 1956), these key breakpoints can be utilized to estimate queue lengths.

Details are avoided here; interested readers are referred to the original study (Liu et al., 2009).

3) Infrastructure-enabled attack detection

Given the estimated queue length from the loop detector and signal status data, we now develop a detector that continuously monitors mobile travel time data and detects data poisoning

attacks. The basic idea is that if the estimated queue length from secure infrastructure data (i.e., loop detector and signal status data) significantly deviates from the one estimated from mobile travel time data, the latter is likely poisoned.

Despite the valuable information the benchmark data provides, it is still challenging to detect attacks due to two issues. First, the queue length estimated from mobile travel time data could deviate from the (typically unknown) true queue length due to measurement errors/noises in the mobile travel time data or systematic bias in the estimation model due to limited penetration of the mobile travel time data. It could be hard to separate such error-induced deviation from the deviation resulting from attacks. Second, the benchmark model and data could also contain biases or errors. An inconsistency with the estimate from the benchmark data may not necessarily be the outcome of a data poisoning attack. Therefore, we propose to train and utilize a machine learning-based model as a detector to differentiate the inconsistency caused by data errors (or model biases) from that induced by data poisoning attacks.

Similar to detecting GPS spoofing, we formulate it as a real-time anomaly detection problem, containing two parts: 1) generating real-time, secure features, and 2) building a machine learning model that determines the queue length estimated from mobile travel time data is anomalous or not given the features computed for signal cycle *k*. Each part is designed as follows.

- *Generating secure features*

As noted earlier and shown in the results, data poisoning attacks can be stealthy. As a result, the conventional way of detecting attacks, which uses the compromised dataset itself for attack detection, is no longer effective. Specifically, features generated from mobile travel time data alone could be compromised, making them less effective in detecting attacks. Therefore, we

leverage the independent, secure loop detector data and signal state data to design a more robust attack detector.

Note that loop detector data and the (sampled) mobile travel time data are not directly relevant, since they are two different data collection processes and represent distinct traffic state information. Thus, we cannot directly use the former for attack detection. Instead, both types of data are processed to estimate queue lengths, which are then compared.

One straightforward feature can be created by computing the difference between the two queue length estimates at k:

$$A^k = Q_d^k - Q_m^k$$

Here, Q_d^k and Q_m^k are the queue length estimated from loop detector data and mobile travel time data for signal cycle k, respectively. In practice, both Q_d^k and Q_m^k contain errors that may be due to the measurement noises/errors in loop detector data and mobile travel time data. As a result, a large deviation between the two estimates (i.e., a large A^k) could result from measurement noises/errors instead of data attacks. Therefore, we take a similar technique as detecting GPS spoofing to improve the robustness of attack detection—applying a sliding window to account for the temporal pattern. Specifically, the features at the previous signal cycles (i.e., A^{k-1}, A^{k-2} ...) are considered when detecting potential attacks at time k. This assumes that the measurement noises/errors are random, and thus it is unlikely to observe multiple consecutive outliers caused by measurement noises/errors. This study selects a sliding window of length three based on the validation dataset. A wide sliding window would reduce the false positive errors but at the cost of inducing false negative errors.

- *Building an Isolation Forest as the detector*

The attack detection is treated as a real-time anomaly detection problem, for which we apply an unsupervised machine learning model to learn anomalies from the data.

The previous section shows that the iForest model has multiple strengths over other unsupervised learning methods in attack detection. Here, an attack detector is again built on the iForest model using the generated features at time k.

$$\delta_k = iForest([A^k, A^{k-1}, A^{k-2}]), \ \ \delta_k \in \{-1, 1\}. \tag{68}$$

Here, iForest produces binary outputs. $\delta_k = 1$ indicates the data is under attack, while $\delta_k = -1$ indicates the data is safe. Attack detection with iForest consists of two stages: 1) a training dataset is used to learn the iForest model, and 2) each testing sample is passed through the iForest model, generating a binary output as an indicator of attack status. As noted in the previous section, iForest is easy to train and it does not require a set of labeled attack and benign samples in the training dataset. Details of the iForest model can be found in the previous section and thus omitted here.

It is likely that other types of unsupervised learning methods could also be effective in detecting attacks. Testing on these methods is left as a future study, as the focus of this study is to demonstrate that it is promising to detect data poisoning attacks using secure infrastructure data, with which a simple detector would be effective.

4) Labeling the poisoned vehicle and correction

Once a potential attack is detected, the estimate from the secure infrastructure data Q_d will be used to replace the potentially compromised estimate. Another important advantage of using secure infrastructure data for defending against attacks on TSEP is that we could tag individual vehicles (or mobile devices) as "poisoned" by comparing its data with the benchmark pattern or the secure data. Tagging a poisoned vehicle is useful as a poisoned vehicle identified at one location (e.g., a signalized intersection) implies that data from the same vehicle may be

compromised or should at least be scrutinized more at other locations. Therefore, not only can existing and newly collected secure data help defend against poisoning attacks at the locations where they are deployed, they will also improve network system-wide security with the ability to tag individual vehicles.

Here, we propose a method to tag potentially poisoned vehicles. This task is not trivial as conventional data cleaning techniques are not applicable. Specifically, since the model learned from the poisoned data has been compromised, outliers identified by a data cleaning technique (typically data points having large residuals) could be pristine rather than poisoned. Inspired by techniques from robust statistics that use trimmed versions of the loss function for robust data modeling, this research applies trimmed optimization techniques for regularized linear regression in adversarial settings.

Assume that x_i $(i \in E)$ are the poisoning data points applied by the attacker, a model could be learned from a subset of the dataset N (i.e., the trimmed dataset) that excludes the poisons. In the ideal case, we would like to find all these poisoning data points and learn the TSEP model using pristine data points only (i.e., $N \backslash E$). However, separating the pristine data points and poisoning points is difficult. To address this issue, in practice, the proposed defense identifies a set of data points with the smallest residuals, which essentially is to solve the following problem.

$$\begin{aligned} &\min_{x_I} L(x_I, \bar{y}). \\ &s.t. \quad I \subset N \end{aligned} \tag{69}$$

Here, L is the loss function and x_I indicates the pristine-like data points (a subset of the data that is unlikely manipulated by the attacker). Note that I could include poisoned points, but only those "similar" to the pristine points, which are thus not expected to contribute much to poisoning the

model. Let $\bar{y}$ represents the learned TSEP representation (i.e., parameters learned from the LS regression for queue length estimation) if all the data points were free from attacks.

From Equation **Error! Reference source not found.**, if the pristine parameter $\bar{y}$ was known, x_I could be easily identified by taking the points having the lowest residual relative to $\bar{y}$. However, in the adversarial setting, $\bar{y}$ is unknown, making it challenging to identify x_I. To address this issue, we use the benchmark model $y^{bestFit}$ from the secure infrastructure data to replace $\bar{y}$. And Equation **Error! Reference source not found.** is modified as the following.

$$\min_{x_I} L\left(x_I, y^{bestFit}\right). \quad s.t. \quad I^{bestFit} \subset N \tag{70}$$

The modified problem is to identify a subset of the dataset that would lead to the estimate closest to that from infrastructure-enable secure data. Though such an estimate from the infrastructure data itself does not necessarily give the ground truth (using $\bar{y}$) due to the measurement errors and biases in the estimation model, we assume that the two are close.

Inspired by the Random Sample Consensus (RANSAC) algorithm, we estimate $y^{bestFit}$ from random samples of potentially poisoned data. The basic assumption is that the observations in a signal cycle k still contain a subset of clean data, and the queue length estimated from this clean data subset is close to that estimated using loop detector data. The sampling method achieves its goal (of tagging potentially attacked vehicles) by following the algorithm in Figure 35.

Once having the best-fit model $y^{bestFit}$, we tag points (vehicles) that have large residuals to $y^{bestFit}$ (exceeding a predefined threshold) as potentially poisoned. Again, note that these tagged are not confirmed poisoned vehicles but shall be scrutinized more at other locations.

Algorithm 2. Tagging attacked vehicles at signal cycle k
Given:

The potentially poisoned data x.

Queue length estimated from loop detector data Q_d.

Return:

bestFit (model $y^{bestFit}$, subset of data $I^{bestFit}$)

Initialization:

Iterations: 0

bestFit: $y^{bestFit} = null,\ I^{bestFit} = null$

$bestErr$ = something really large

Repeat (if iteration $s < maxIter$):

1. Select a random subset of the data x_{I^s}.
2. Based on x_{I^s}, a model y^s is fitted.
3. The queue length Q_{I^s} is estimated.
4. Compare Q_{I^s} with Q_d and update bestFit model is needed.

   if $abs(Q_{I^s} - Q_d) < bestErr$:

   $y^{bestFit} = y^s$

   $I^{bestFit} = I^s$

   $bestErr = abs(Q_{I^s} - Q_d)$

   increment iterations: $s = s + 1$.

Return: $y^{bestFit}$

Figure 35. Algorithm for tagging attacked vehicles.

5.2.4. Results from defending against attacks on queue length estimation model

1) Experiment setting

We start with some simple simulations to test the proposed infrastructure-enabled solution to defend against attacks on the queue length estimation model. The test simulates an intersection with a fixed signal configuration. Two data collection processes are simulated using SUMO, including 1) sampling mobile travel time data at VLTs and 2) acquiring detector loop data and signal state data. SUMO simulation can also output the true queue length in each signal cycle,

which helps to evaluate the impact of attacks and the mitigation of the proposed defense. In order to launch data attacks, travel time data in some signal cycles are modified following the attack model presented in Section 4.4.

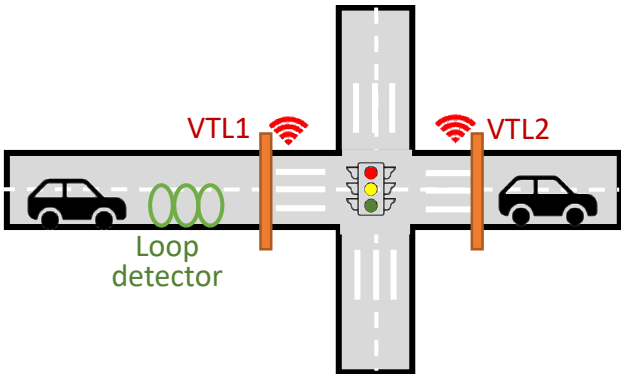

Figure 36. SUMO simulation for defending against data poisoning attacks.

To collect mobile travel time data, a sample of vehicles is randomly selected, and their trajectories are recorded during the simulation. Then, two VTLs are set upstream and downstream of the West-East approach. Each sampled vehicle's arrival times at the two locations can be determined from its trajectory, which can be used to compute the travel time and the delay at the intersection. Such information allows for constructing delay patterns at the intersection (Figure 37) for estimating queue length following the study in (Ban et al., 2011). The data is collected wirelessly and can be vulnerable to data poisoning attacks, as demonstrated in Section 4.4.

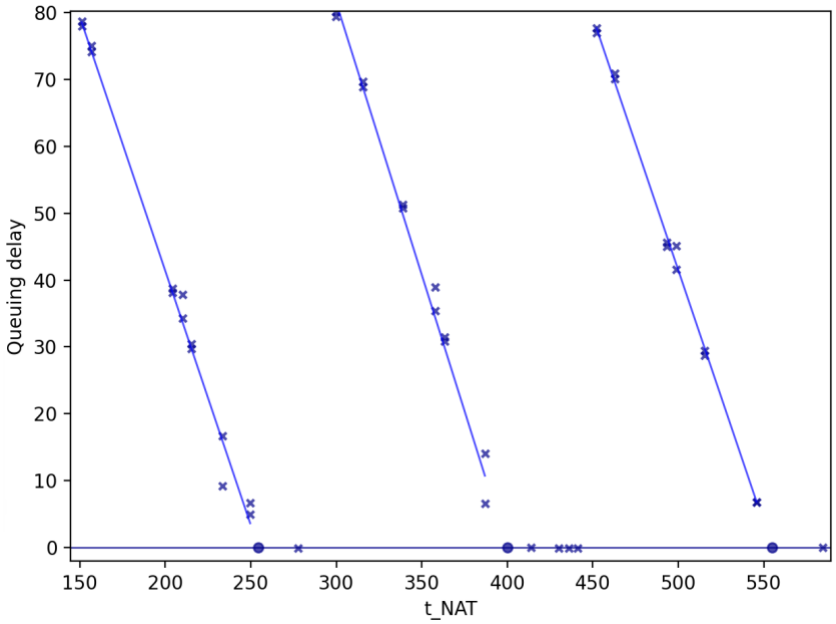

Figure 37. Estimating queue length using mobile travel time data simulated from SUMO.

In order to acquire detector loop data, a detector is configured for the WE approach, positioned upstream of the intersection. It generates the entry and exit times of each vehicle passing over the detector. The occupancy time and gap time of two consecutive vehicles can be computed naturally (Figure 38). As noted earlier, these two metrics and signal status data are crucial in estimating queue lengths. The signal status data can be derived easily from the signal schedule, which is predefined and implemented by SUMO. Figure 39 compares the queue length estimates from loop detector data and the direct SUMO outputs (ground truth) for each cycle. It can be observed that while the benchmark estimates (free from attacks) are very close to the true values, deviations do exist due to measurement errors and biases in the estimation model.

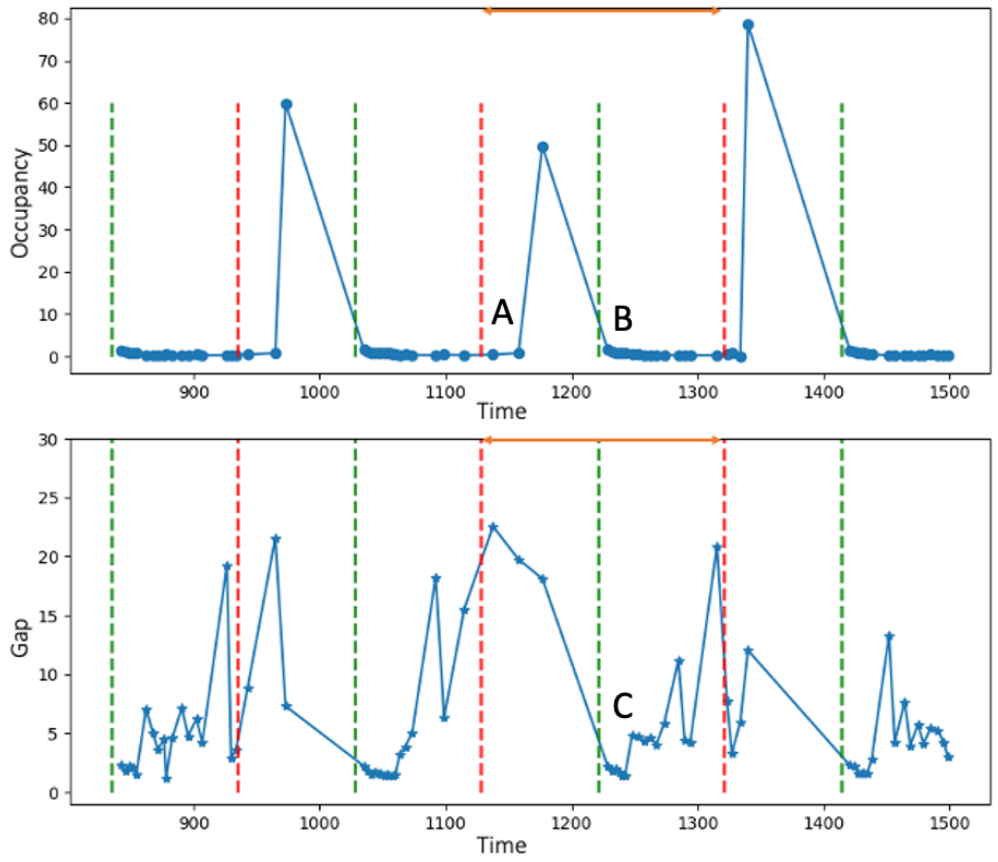

Figure 38. Demonstration of estimating queue length using detector data (occupancy and time gap)

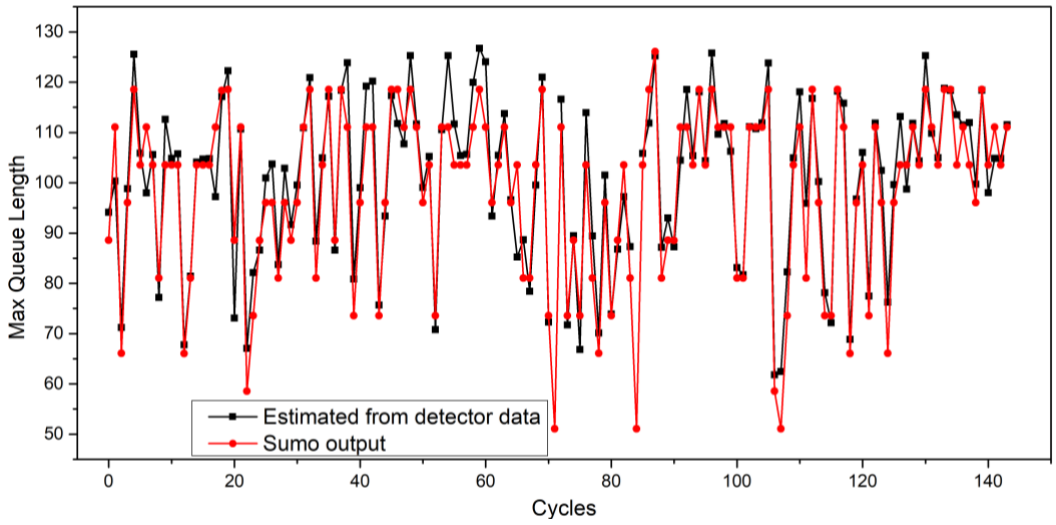

Figure 39. Comparison of queue lengths estimated from loop detector data and SUMO output (ground truth).

To launch a data poisoning attack, we randomly select one signal cycle and perturb travel time data (of some vehicles) in the cycle following the attack model developed in Section 4.4.1. Both types of data poisoning attacks are simulated, including the perturbation-type and addition-type attacks (see Section 4.4.1 for more details).

Since the true queue length and the poisoned travel time data are known in the simulation tests, we can evaluate the effectiveness of the proposed defense method. The defense performance is evaluated from multiple perspectives:

i. If a data poisoning attack is launched, the defense should be able to detect it and raise the alert.
ii. The poisoned vehicles should be tagged.
iii. The queue length gets corrected effectively following the proposed method.

We then conduct sensitivity analyses to investigate the effects of several factors, including measurement errors in mobile travel time data (the data itself) and the hyperparameter of the iForest model (the attack detection model).

Following the previous tests on defending GPS spoofing, we adopt several metrics to measure the detection accuracy, including the *F1* score, *precision*, and *recall*. *Mean Absolute Percent Error* (*MAPE*) measures the errors in queue length estimations before and after detecting the attacks and correcting the estimates following the proposed defense. Specifically, the MAPE is computed by comparing the queue length estimate with the one produced by SUMO (i.e., ground-truth estimate) at each signal cycle and then evaluating the average value of all cycles.

$$MAPE = \frac{1}{K}\sum_{k}\frac{\left|Q_m^k - Q_d^k\right|}{Q_d^k} \times 100$$

2) Testing results

Figure 40 compares the queue lengths estimated using mobile travel time data Q_m and those using loop detector data Q_d without attacks. We can observe that even though there is no attack, the estimates from the two data sources can be different, which could be due to either the erroneous measurements or the inaccuracy of the models. Such differences make it challenging for attack detection, as the attack detector would mistake the deviation of Q_m from Q_d due to errors as an attack, leading to false positive errors.

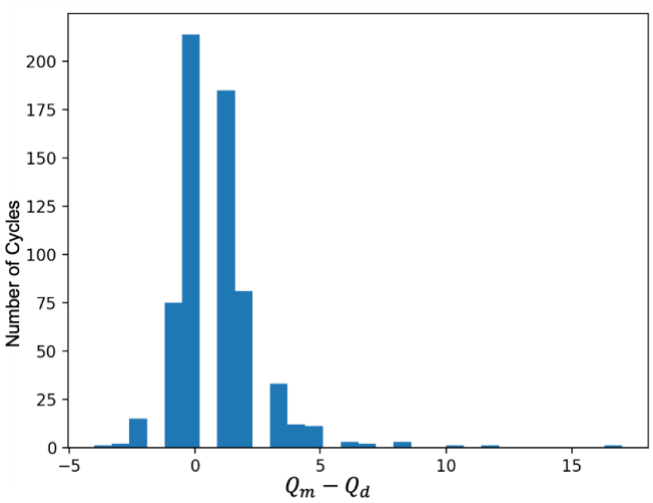

Figure 40. Comparison of queue lengths estimated from clean mobile travel time data and loop detector data.

- Detection accuracy

We randomly select and attack a few signal cycles to test the detection accuracy. The attack detector examines each signal cycle to detect possible attacks. Table 6 reports the F1 score, precision and recall as measures of the detection accuracy. Overall, it can be observed that the attack detector based on infrastructure data works in identifying attacks but could be improved further. The improvement could be made from several aspects. First, the accuracy of the benchmark queue length estimated from the infrastructure data should be improved by either

reducing the measurement errors in data or mitigating errors in the estimation model. This treatment will likely reduce the challenge of differentiating the attacks from the deviations resulting from inaccurate benchmark data. Second, the attack detector should be further fine-tuned such that it can distinguish the TSEP solution deviation due to random errors in data from that due to poisoning attacks. The fine-tuning process is particularly necessary if the attacks are stealthy while the random errors are significantly large.

Table 6. Performance of detecting attacked queue lengths

	F1 score	Precision	Recall
Detection accuracy	0.59	0.58	0.6

- The queue length gets corrected

As noted earlier, when an attack is detected at a signal cycle, the queue length estimate from mobile travel time data is replaced by that from loop detector data. To evaluate the performance of such correction in terms of improving queue length estimation, we compute MAPE of queue length estimates without and with the correction. The results are reported in Table 7. We can observe that the MAPE can be improved by 22% with detection and correction.

Table 7. RSME of estimated queue length with/without considering detection and correction

	Without detection and correction	With detection and correction	Reduction
MAPE of queue length estimation	18	14	-22%

- Poisoned vehicles should be identified

Table 8 reports the accuracy of tagging attacked vehicles. In general, it shows that the accuracy is low. A recall of 0.21 suggests that about one-fifth of poisoned vehicles are successfully

tagged. This could be because of the stealthiness of the attack model: the perturbations added to the (attacked) vehicles are small, making it challenging to differentiate the poisons from the measurement noises. Also note that under the proposed attack model, only a few vehicles are poisoned, so the dataset becomes highly imbalanced. Such imbalance adds another layer of challenges to identifying the attacked vehicles, demanding fine-tuning of the attack detector to improve the detection accuracy.

Table 8. Performance of tagging attacked vehicles

	F1 score	Precision	Recall
Accuracy of tagging attacked vehicles	0.26	0.32	0.21

3) Comparing with other defense solutions

Here, the proposed IED solution is compared with other existing defenses from the literature, including Huber regression, RANSAC, and TRIM. Huber regression (Huber, 1964) and RANSAC (Fischler and Bolles, 1981) are two well-known methods in robust statistics; both are robust to outliers in the data. TRIM is a recently developed defense against data poisoning attacks on regression models (Jagielski et al., 2018).

Huber regression is an alternative to least-squared regression that is sensitive to outliers. Huber regression introduces the Huber loss function to limit the contribution of outliers to computing the loss. Specifically, the Huber loss function applies quadratic terms for points with small residuals and linear terms for points with large residuals. The two groups of points are separated with a predetermined threshold.

RANSAC (or random sample consensus) is an iterative method to estimate a mathematical model from a data set that contains outliers. It works by identifying the outliers in a data set and

estimating the desired model using data that does not contain outliers. To learn a linear model excluding outliers, RANSAC uses repeated random sub-sampling and in each repeat, a linear model is fitted on the random sub-sample of data. A basic assumption is that the data consists of inliers and outliers, and a procedure exists to well estimate the parameters of the desired model from a (usually small) set of inliers. During the iterative process, a model is rejected if there are too many outliers according to the fitted model and a new model is estimated from a different random sub-sample of data. The specific model rejection criterion is tuned by predefined hyperparameters.

TRIM algorithm is a recent development in the adversarial machine learning field (Jagielski et al., 2018). It has been demonstrated that TRIM outperforms other defenses on a range of models and real-world datasets (Jagielski et al., 2018). TRIM is similar to algorithms from robust statistics that apply trimmed versions of the loss function but is specifically improved for training a regression model with poisoned data. Instead of removing outliers before fitting a model, TRIM iteratively estimates regression parameters, where in each iteration, the model is fitted on a subset of data samples with the lowest residuals. Therefore, TRIM regulates the regression in the adversarial setting by applying trimmed optimization techniques (i.e., computing a trimmed loss function based on a different subset of residuals in each iteration).

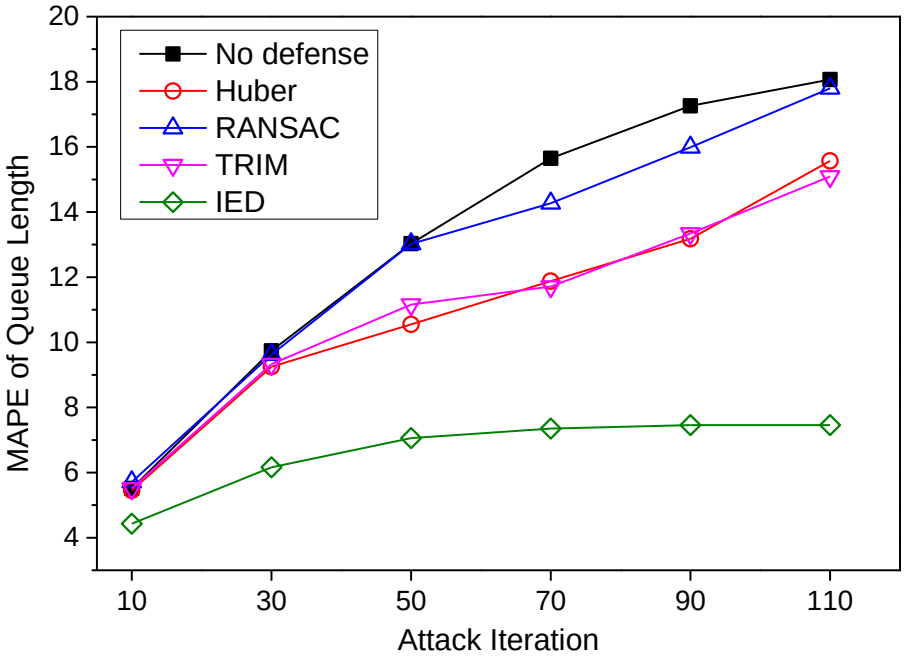

Figure 41. Comparison of defenses against data poisoning attacks on queue length estimation.

Figure 41 compares the performance of the various defense solutions under data poisoning attacks, including the existing ones and the proposed IED solution. The performance is evaluated by averaging the results from over a hundred signal cycles. The baseline gives the mean absolute percent errors (MAPE) of queue length estimations without any defense under the different numbers of attack iterations. A larger number of attack iterations means more perturbations added in the data, leading to a larger MAPE in estimated queue lengths. MAPE is computed by comparing the results with those from the pristine model. Figure 41 provides several observations as below.

- *The data poisoning attack (proposed in Section 4) could fail the (tested) existing defense solutions.*

  As seen in Figure 41, existing defenses could not effectively mitigate the impacts of poisoning attacks. With more perturbations added to the data, these defenses cannot guard the learning model as the MAPE of queue lengths increases significantly. RANSAC, in

particular, hardly reduces MAPE over the undefended model. The other two defenses (e.g., Huber and TRIM) show similar performance: both of them are unable to mitigate the adversarial impacts under subtle poisons (e.g., attack iterations are 10 and 30); they can reduce MAPE significantly under large poisons (e.g., attack iterations larger than 50) in which cases these poisons are more distinguishable from the subtle poisons.

- *The proposed IED solution is more effective in defending against the data poisoning attack than the existing defenses.*

  Figure 41 shows clear advantages of the IED solution over the existing defenses. First, under all the attack scenarios, the IED solution further mitigates the MAPE of queue length. Second, by applying the IED solution, MAPE is more controlled instead of increasing wildly as more adversarial perturbations are added to the data: MAPE increases by 2.5 (from 4.5 to 7) when the number of attack iterations increases from 10 to 110. Again, this confirms the benefit of using secure infrastructure data as an independent data source for attack detection and correction. Ideally, MAPE following the IED solution should be independent of the number of attack iterations given the independent, secure infrastructure data. However, this is challenging due to the false negative errors, which means that the detector misses attacks on some signal cycles and, consequently, the compromised queue length estimations would not be corrected.

4) Sensitive analysis

Multiple factors are related to the implementation of the infrastructure-enabled method. They could affect the performance of the proposed method either directly or indirectly. We conduct sensitivity analyses of the infrastructure-enabled method in this subsection on these factors.

- The effect of measurement errors in mobile travel time data

Our previous investigation indicates that the queue length estimation could be inaccurate due to measurement errors in mobile travel time data. Therefore, Q_m could deviate Q_d because of either the errors in the data or data poisoning attacks. A scenario with large measurement errors could challenge distinguishing attacks from data's inherent errors. In order to investigate this effect, we create different scenarios by adding different levels of random errors (norm distributions of different standard deviations) to the original data generated from SUMO simulations. Table 9 compares the performance of the proposed solution under two levels of measurement errors. We can observe that with large measurement errors in the data, the accuracy of detecting the attacks and tagging manipulated vehicles drops significantly. Meanwhile, the MAPE of estimated queue lengths increases and the improvement of estimates via the correction is limited by 9%.

Table 9. Effect of measurement errors

		Error = 3	Error = 5
Detection Accuracy	F1 score	0.59	0.33
	precision	0.58	0.28
	Recall	0.6	0.41
MAPE	Before correction	18	28
	After correction	14 (-22%)	25 (-9%)
Tagging Vehicle Accuracy	F1 score	0.26	0.19
	precision	0.32	0.19
	Recall	0.21	0.18

- The effect of iForest settings

In the proposed infrastructure-enabled solution method, iForest is the core algorithm for attack detection. One key hyperparameter of iForest is *contamination* (denoted as α), which specifies the proportion of spoofed samples in the data set. We investigate the sensitivity of the infrastructure-enabled method with α.

Table 10 summarizes the performance of the proposed method with different α's. It can be observed that with a higher α, the recall is reduced, suggesting that the iForest detector is less sensitive in identifying the attacks. Though it is important to balance the false-positive and the false-negative rates, in our scenario a high false positive rate is preferable. This is because the risk of missing poisoned vehicles is high, while falsely tagged vehicles could be corrected with additional data at other intersections. Therefore, in our study, a small α is preferred to yield a high recall that reduces false-negative errors.

Table 10. Effect of iForest settings (hyper-parameter α)

		$\alpha = 0.1$	$\alpha = 0.2$
Detection Accuracy	F1 score	0.59	0.56
	Precision	0.58	0.70
	Recall	0.60	0.46
MAPE	Before correction	18	18
	After correction	14 (-22%)	14 (-22%)
Tagging vehicle Accuracy	F1 score	0.26	0.21
	Precision	0.32	0.39
	Recall	0.21	0.15

5.2.5. Summary

This section uses the queue length estimation model to illustrate the idea of infrastructure-enabled defenses against data attacks on TSEP. A detection and correction framework is developed. It uses infrastructure status data and existing or newly deployed secure data from the infrastructure (e.g., loop detector data) to detect attacks on the queue length estimation model. Specifically, the detection relies on computing features from the benchmark data from wired loop detectors and the signal controller. Based on these features, an Isolation Forest model is trained and evaluated in real-time to detect data attacks. Once an attack is detected, the potentially compromised queue

length is corrected using the one computed from loop detector data. Additionally, the potentially attacked vehicle could be tagged as a means to monitor the credibility of vehicles on the road. The simulation tests demonstrate the effectiveness of the proposed IED solution. That is to use the secure data from existing infrastructure as the benchmark to defend against data poisoning attacks. Comparing with the existing defense solutions (e.g., Huber, RANSAC and TRIM), the proposed IED solution is more effective in mitigating the errors in queue length estimates due to the attacks. The advantage is consistently observed under both subtle and heavy poison attacks. In addition, the sensitivity of the proposed solution to several key factors is explored. The results show that the performance of the solution could be sensitive to the data accuracy: the accuracy of detecting and tagging attacked vehicles could be reduced if the measurement noises in the loop detector data and mobile travel time data are large, which renders it challenging to differentiating data (inherent) errors from data attacks.

6. DISCUSSION AND FUTURE RESEARCH

The future work of this dissertation research will mainly focus on the following aspects: 1) Extending the study of data poisoning attacks against more TSEP models; 2) Improving the infrastructure-enabled defense methods for TSEP models.

6.1. EXTENSION OF DATA POISONING ATTACKS AGAINST THE TSEP MODEL

This study focuses on attacking optimization-based TSEP models with an explicit objective function. In practice, these models could be complex and/or have objective functions that cannot be expressed explicitly.

The training (and sometimes inference) of many statistical learning models rely on optimization techniques but the models could have complex objective functions. One such model is Bayesian networks (BN), which has been used for (cycle by cycle) vehicle index and queue length (distribution) estimation (Hao et al., 2014), human mobility modeling (Toch et al., 2019), etc. The objective functions of the estimation models in statistical learning can be complex, which may bring difficulties to model attacks and solutions. One promising direction is to investigate methods to approximate complex functions that are from the objective functions or constraints of the attack models on statistical learning.

For deep learning (DL) models, the target model can be even more complex and often unknown, as the attacker does not necessarily know the model's key components, such as the training data, model structure, or parameters. In such a case, the promising idea is to develop the attack model based on the transferability of DL models rather than directly working on the target model (Xie et al., 2022b). The transferability ensures that the derived attack model from public data and DL structures can be transferred and applied to attack other DL-based TSEP models.

Multiple solutions may exist for either the statistical learning or DL-based TSEP models. This implies that $P(x)$ in Equation (1) is now a set-valued map and the second Lipschitz continuity definition in Section 2.3 should apply. The subsequent analysis and the attack models should also be constructed accordingly (i.e., based on set-valued maps rather than assuming that $P(x)$ is a singleton).

6.2. ENHANCING DEFENSE SOLUTIONS

As mentioned earlier, there are two approaches to defending against data poisoning attacks: the "security by design" approach and the detection and mitigation approach. This dissertation follows the second approach while leaving the first one as future research, which could exercise the security by design principle to devise data poisoning defenses following the Lipschitz analysis. For a TSEP model, if the model solution turns out to be not Lipschitz continuous with data perturbations, one needs to explore ways (to reformulate or approximate the objective/constraints or even the structure of the model) to revise/refine the model so that the Lipschitz property may hold. Doing so will make the model more robust to data attacks (i.e., making the TSEP solution less sensitive to poisons). In fact, in the DL literature, research has been conducted to estimate the Lipschitz constants of DL models (Scaman and Virmaux, 2019) to assess the robustness of such models. One can also use such estimated constants to assess the resilience of the model w.r.t. data poisoning attacks. What is missing in the literature is that if the model turns out to be not robust (or not resilient), what methods can be used to secure the model. Future research should give more focus on this regard.

The proposed infrastructure-enabled defense method follows the detection and mitigation approach, which has limitations and can be improved. First, more advanced learning approaches (such as deep learning) may enhance the detection and correction methods to improve their

performances. Second, more research efforts are needed to design optimal strategies for deploying the roadside infrastructure for cybersecurity purposes. In the solution proposed to defend against GPS spoofing, it is assumed that RSUs are deployed evenly on the roadside and conduct sensitivity analysis to understand the impact of the distance between two consecutive RSUs on the IED's performance. For future research, the optimal RSU deployment problem may be studied to produce RSU deployment strategies that systematically consider the deployment cost, traffic environments, road geometry, and the performance of the spoofing defense method. Third, future investigations are needed to test the IED's performance in real-world driving scenarios. Fourth, the IED method may be enhanced by improving the benchmark data or incorporating additional (and easily obtained) data sources for more robust attack detection. This is particularly so for GPS spoofing scenarios where the distance between RSUs is large. For instance, the geometric outlines of roads may be used as constraints to improve location estimation/prediction (Kaiwartya et al., 2018), which may further improve detection accuracy. It has also shown that the accuracy of loop detector data could affect the performance of defending against attacks on the queue length estimation model. Last but not least, as the infrastructure is becoming more important to support advanced vehicle technologies and systems, the idea of the proposed IED method may also be applied to detecting and mitigating other types of data attacks in transportation. This may include, e.g., adding adversarial images to onboard cameras (Xie et al., 2022b) or spoofing attacks on LiDAR data (Cao et al., 2019), which should be formally studied.

7. CONCLUSIONS

This dissertation focuses on the data poisoning attack, one of the emerging cybersecurity issues in transportation. It first develops a generic data poisoning attack model for traffic state estimation and prediction (TSEP) applications that primally rely on infrastructure-generated data. Then a general framework is presented to analyze the Lipschitz properties of the TSEP solution with respect to the data. The framework investigates the parameterized version of a nonlinear optimization problem, where the aim is to study the behavior of the TSEP solution under the perturbation of data (Dontchev and Rockafellar, 2009). This dissertation uses a generalized implicit function theorem to develop a poisoning attack algorithm t. The proposed framework is then tailored to typical TSEP models, including the LS regression model and SVM. Experiments are conducted using real-world models and datasets. The results suggest that common TSEP models could be vulnerable to the proposed data poisoning attacks.

This dissertation then develops defense solutions to data poisoning attacks. Recognizing the stealthiness of data poisoning attacks, it proposes a novel defense solution using secure data from transportation infrastructure, which is a recent parallel development for supporting various functionalities of advanced vehicle technologies. By securing the data collection and transmission processes, the independent, secure data from the infrastructure enables a promising way to defend against data poisoning attacks. This dissertation utilizes such secure data and develops infrastructure-enabled solutions to defend against attacks on vehicular data and TSEP models. To demonstrate the effectiveness of the proposed solution, it first tests the solution in defending against GPS spoofing, a high threat of advanced vehicle driving systems. The design of infrastructure-enabled GPS spoofing detection and correction is presented, followed by experimental tests using simulation and real-world GPS data, under various scenario settings and

spoofing attacks. The effectiveness of the infrastructure-enabled solution is evaluated by comparing it with state-of-the-art GPS spoofing defense strategies. The infrastructure-enabled defense against attacks on TSEP is then developed and evaluated using the queue length estimation model as the case study, which is shown to be vulnerable to data poisoning attacks. Simulation tests are conducted, and the results demonstrate the performance of the proposed solution.

This dissertation also discusses the limitations of the developed methods in this dissertation and identifies future research directions. For example, this proposed attack model does not apply to TSEP models that cannot be expressed via explicit formulas. Future work will extend it to more general TSEP models, such as the statistical-learning-based or deep learning-based models that have either complex or implicit objective functions only. The limitations of the defense solutions to GPS spoofing (e.g., the optimal deployment of RSUs that provide secure benchmark data) and attacks on TSEP models (e.g., more advanced machine-learning-based attack detectors) are also discussed; promising ideas to mitigate these limitations are listed. Again one of the motivations of this research is to draw the attention of the transportation community to the cybersecurity issues in transportation and initiate the community's efforts to tackle the emerging challenges, especially to develop innovative methods that are based on future emerging intelligent transportation infrastructure systems.

BIBLIOGRAPHY

Abdollahi Biron, Z., Dey, S., Pisu, P., 2018. Real-Time Detection and Estimation of Denial of Service Attack in Connected Vehicle Systems. IEEE Trans. Intell. Transp. Syst. 19, 3893–3902. https://doi.org/10.1109/TITS.2018.2791484

Abt, N., 2018. The cybersecurity battle [WWW Document]. FleetOwner. URL https://www.fleetowner.com/technology/article/21702640/the-cybersecurity-battle (accessed 4.3.22).

Adegoke, E.I., Zidane, J., Kampert, E., Ford, C.R., Birrell, S.A., Higgins, M.D., 2019. Infrastructure Wi-Fi for connected autonomous vehicle positioning: A review of the state-of-the-art. Veh. Commun. 20, 100185. https://doi.org/10.1016/j.vehcom.2019.100185

Ahmad, F., Adnane, A., Franqueira, V.N.L., 2016. A Systematic Approach for Cyber Security in Vehicular Networks. J. Comput. Commun. 04, 38–62. https://doi.org/10.4236/jcc.2016.416004

Alawatugoda, J.A.B., 2015. On the leakage resilience of secure channel establishment.

Alnasser, A., Sun, H., Jiang, J., 2019. Cyber security challenges and solutions for V2X communications: A survey. Comput. Netw. 151, 52–67. https://doi.org/10.1016/j.comnet.2018.12.018

Ban, X., Gruteser, M., 2012. Towards fine-grained urban traffic knowledge extraction using mobile sensing. Presented at the Proceedings of the ACM SIGKDD International Workshop on Urban Computing, pp. 111–117.

Ban, X. (Jeff), Hao, P., Sun, Z., 2011. Real time queue length estimation for signalized intersections using travel times from mobile sensors. Transp. Res. Part C Emerg. Technol. 19, 1133–1156. https://doi.org/10.1016/j.trc.2011.01.002

Ban, X. (jeff, Herring, R., Bayen, A.M., Hao, P., 2009. Delay pattern estimation for signalized intersections using sampled travel times. Transportation Research Record, in: Transportation Research Board of the National Academies.

Bellare, M., Rogaway, P., 2005. Introduction to modern cryptography. Ucsd Cse 207, 207.

Bey, T., Tewolde, G., 2019. Evaluation of DSRC and LTE for V2X, in: 2019 IEEE 9th Annual Computing and Communication Workshop and Conference (CCWC). Presented at the 2019 IEEE 9th Annual Computing and Communication Workshop and Conference (CCWC), pp. 1032–1035. https://doi.org/10.1109/CCWC.2019.8666563

Biggio, B., Nelson, B., Laskov, P., 2013. Poisoning Attacks against Support Vector Machines. ArXiv12066389 Cs Stat.

Biggio, B., Roli, F., 2018. Wild patterns: Ten years after the rise of adversarial machine learning. Pattern Recognit. 84, 317–331. https://doi.org/10.1016/j.patcog.2018.07.023

Bishop, C., 2006. Pattern Recognition and Machine Learning, Information Science and Statistics. Springer-Verlag, New York.

Boeira, F., Asplund, M., Barcellos, M.P., 2018. Vouch: A Secure Proof-of-Location Scheme for VANETs, in: Proceedings of the 21st ACM International Conference on Modeling, Analysis and Simulation of Wireless and Mobile Systems. Presented at the MSWIM '18: 21st ACM Int'l Conference on Modelling, Analysis and Simulation of Wireless and Mobile Systems, ACM, Montreal QC Canada, pp. 241–248. https://doi.org/10.1145/3242102.3242125

Boriboonsomsin, K., Vu, A., Barth, M., 2010. Eco-driving: pilot evaluation of driving behavior changes among us drivers. University of California Transportation Center. UCTC-FR-2010-20.

Boyd, S., Vandenberghe, L., 2004. Convex Optimization. Cambridge University Press.

Brecht, B., Therriault, D., Weimerskirch, A., Whyte, W., Kumar, V., Hehn, T., Goudy, R., 2018. A Security Credential Management System for V2X Communications. Trans Intell Transp. Sys 19, 3850–3871. https://doi.org/10.1109/TITS.2018.2797529

Brumback, B., Srinath, M., 1987. A Chi-square test for fault-detection in Kalman filters. IEEE Trans. Autom. Control 32, 552–554. https://doi.org/10.1109/TAC.1987.1104658

Burges, C.J.C., 1998. A Tutorial on Support Vector Machines for Pattern Recognition. Bell Lab. 43.

Callahan, J., 2019. Transportation Cybersecurity Incidents.

Cao, Y., Xiao, C., Cyr, B., Zhou, Y., Park, W., Rampazzi, S., Chen, Q.A., Fu, K., Mao, Z.M., 2019. Adversarial Sensor Attack on LiDAR-based Perception in Autonomous Driving. Proc. 2019 ACM SIGSAC Conf. Comput. Commun. Secur. 2267–2281. https://doi.org/10.1145/3319535.3339815

Chen, Z., Heckman, C., Julier, S., Ahmed, N., 2018. Weak in the NEES?: Auto-tuning Kalman Filters with Bayesian Optimization. ArXiv180708855 Cs Eess Stat.

Cui, Z., 2021. Deep Learning for Short-Term Network-Wide Road Traffic Forecasting. University of Washington.

Dadras, S., Gerdes, R.M., Sharma, R., 2015. Vehicular Platooning in an Adversarial Environment, in: Proceedings of the 10th ACM Symposium on Information, Computer and Communications

Security. Presented at the ASIA CCS '15: 10th ACM Symposium on Information, Computer and Communications Security, ACM, Singapore Republic of Singapore, pp. 167–178. https://doi.org/10.1145/2714576.2714619

Diehl, C.P., Cauwenberghs, G., 2003. Svm incremental learning, adaptation and optimization, in: Proceedings of the International Joint Conference on Neural Networks, 2003. Presented at the International Joint Conference on Neural Networks, 2003., IEEE, Portland, Oregon USA, pp. 2685–2690. https://doi.org/10.1109/IJCNN.2003.1223991

Ding, Z., Fei, M., 2013. An Anomaly Detection Approach Based on Isolation Forest Algorithm for Streaming Data using Sliding Window. IFAC Proc. Vol. 46, 12–17. https://doi.org/10.3182/20130902-3-CN-3020.00044

Dissanayake, G., Sukkarieh, S., Nebot, E., Durrant-Whyte, H., 2001. The aiding of a low-cost strapdown inertial measurement unit using vehicle model constraints for land vehicle applications. IEEE Trans. Robot. Autom. 17, 731–747. https://doi.org/10.1109/70.964672

Dontchev, A.L., Rockafellar, R.T., 2009. Implicit functions and solution mappings. Springer.

Feng, Y., Huang, S., Chen, Q.A., Liu, H.X., Mao, Z.M., 2018. Vulnerability of Traffic Control System Under Cyberattacks with Falsified Data. Transp. Res. Rec. J. Transp. Res. Board 2672, 1–11. https://doi.org/10.1177/0361198118756885

Ferrari Dacrema, M., Cremonesi, P., Jannach, D., 2019. Are we really making much progress? A worrying analysis of recent neural recommendation approaches, in: Proceedings of the 13th ACM Conference on Recommender Systems, RecSys '19. Association for Computing Machinery, New York, NY, USA, pp. 101–109. https://doi.org/10.1145/3298689.3347058

Fiacco, A.V., 1983. Introduction to Sensitivity and Stability Analysis in Nonlinear Programming. Academic Press.

Fischler, M.A., Bolles, R.C., 1981. Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. Commun. ACM 24, 381–395. https://doi.org/10.1145/358669.358692

Gao, Y., Liu, S., Atia, M.M., Noureldin, A., 2015. INS/GPS/LiDAR Integrated Navigation System for Urban and Indoor Environments Using Hybrid Scan Matching Algorithm. Sensors 15, 23286–23302. https://doi.org/10.3390/s150923286

Greenberg, A., 2015. Hackers Remotely Kill a Jeep on the Highway—With Me in It | WIRED [WWW Document]. URL https://www.wired.com/2015/07/hackers-remotely-kill-jeep-highway/ (accessed 4.3.22).

Greer, L., Fraser, J.L., Hicks, D., Mercer, M., Thompson, K., 2018. Intelligent transportation systems benefits, costs, and lessons learned: 2018 update report. United States. Dept. of Transportation. ITS Joint Program Office.

Guerrero-Higueras, Á.M., DeCastro-García, N., Matellán, V., 2018. Detection of Cyber-attacks to indoor real time localization systems for autonomous robots. Robot. Auton. Syst. 99, 75–83. https://doi.org/10.1016/j.robot.2017.10.006

Guerrero-Ibáñez, J., Zeadally, S., Contreras-Castillo, J., 2018. Sensor Technologies for Intelligent Transportation Systems. Sensors 18, 1212. https://doi.org/10.3390/s18041212

Guo, Q., Angah, O., Liu, Z., Ban, X. (Jeff), 2021. Hybrid deep reinforcement learning based eco-driving for low-level connected and automated vehicles along signalized corridors. Transp. Res. Part C Emerg. Technol. 124, 102980. https://doi.org/10.1016/j.trc.2021.102980

Hao, P., Ban, X. (Jeff), Guo, D., Ji, Q., 2014. Cycle-by-cycle intersection queue length distribution estimation using sample travel times. Transp. Res. Part B Methodol. 68, 185–204. https://doi.org/10.1016/j.trb.2014.06.004

Hasan, M., Mohan, S., Shimizu, T., Lu, H., 2020. Securing Vehicle-to-Everything (V2X) Communication Platforms. IEEE Trans. Intell. Veh. 5, 693–713. https://doi.org/10.1109/TIV.2020.2987430

Hofmann-Wellenhof, B., Lichtenegger, H., Wasle, E., 2007. GNSS – Global Navigation Satellite Systems: GPS, GLONASS, Galileo, and more. Springer Science & Business Media.

Huang, L., Joseph, A.D., Nelson, B., Rubinstein, B.I., Tygar, J.D., 2011. Adversarial machine learning. Presented at the Proceedings of the 4th ACM workshop on Security and artificial intelligence, pp. 43–58.

Huang, S.E., Feng, Y., Liu, H.X., 2021. A data-driven method for falsified vehicle trajectory identification by anomaly detection. Transp. Res. Part C Emerg. Technol. 128, 103196. https://doi.org/10.1016/j.trc.2021.103196

Huber, P.J., 1964. Robust Estimation of a Location Parameter. Ann. Math. Stat. 35, 73–101.

Jagielski, M., Oprea, A., Biggio, B., Liu, C., Nita-Rotaru, C., Li, B., 2018. Manipulating Machine Learning: Poisoning Attacks and Countermeasures for Regression Learning, in: 2018 IEEE Symposium on

Security and Privacy (SP). Presented at the 2018 IEEE Symposium on Security and Privacy (SP), IEEE, San Francisco, CA, pp. 19–35. https://doi.org/10.1109/SP.2018.00057

Jiang, W., Li, H., Liu, S., Luo, X., Lu, R., 2020. Poisoning and Evasion Attacks Against Deep Learning Algorithms in Autonomous Vehicles. IEEE Trans. Veh. Technol. 69, 4439–4449. https://doi.org/10.1109/TVT.2020.2977378

Kaiwartya, O., Cao, Y., Lloret, J., Kumar, S., Aslam, N., Kharel, R., Abdullah, A.H., Shah, R.R., 2018. Geometry-Based Localization for GPS Outage in Vehicular Cyber Physical Systems. IEEE Trans. Veh. Technol. 67, 3800–3812. https://doi.org/10.1109/TVT.2018.2796242

Katrakazas, C., Quddus, M., Chen, W.-H., Deka, L., 2015. Real-time motion planning methods for autonomous on-road driving: State-of-the-art and future research directions. Transp. Res. Part C Emerg. Technol. 60, 416–442. https://doi.org/10.1016/j.trc.2015.09.011

Khan, S.K., Shiwakoti, N., Stasinopoulos, P., Chen, Y., 2020. Cyber-attacks in the next-generation cars, mitigation techniques, anticipated readiness and future directions. Accid. Anal. Prev. 148, 105837. https://doi.org/10.1016/j.aap.2020.105837

Khattab, A., Fahmy, Y.A., Abdel Wahab, A., 2015. High Accuracy GPS-Free Vehicle Localization Framework via an INS-Assisted Single RSU. Int. J. Distrib. Sens. Netw. 11, 795036. https://doi.org/10.1155/2015/795036

Koh, P.W., Steinhardt, J., Liang, P., 2021. Stronger Data Poisoning Attacks Break Data Sanitization Defenses. https://doi.org/10.48550/arXiv.1811.00741

Kutila, M., Pyykonen, P., Huang, Q., Deng, W., Lei, W., Pollakis, E., 2019. C-V2X Supported Automated Driving, in: 2019 IEEE International Conference on Communications Workshops (ICC Workshops). Presented at the 2019 IEEE International Conference on Communications Workshops (ICC Workshops), pp. 1–5. https://doi.org/10.1109/ICCW.2019.8756871

Kuutti, S., Fallah, S., Katsaros, K., Dianati, M., Mccullough, F., Mouzakitis, A., 2018. A Survey of the State-of-the-Art Localization Techniques and Their Potentials for Autonomous Vehicle Applications. IEEE Internet Things J. 5, 829–846. https://doi.org/10.1109/JIOT.2018.2812300

Lee, B.-H., Dewi, E.K., Wajdi, M.F., 2018. Data security in cloud computing using AES under HEROKU cloud, in: 2018 27th Wireless and Optical Communication Conference (WOCC). Presented at the 2018 27th Wireless and Optical Communication Conference (WOCC), pp. 1–5. https://doi.org/10.1109/WOCC.2018.8372705

Lei, A., Cruickshank, H., Cao, Y., Asuquo, P., Ogah, C.P.A., Sun, Z., 2017. Blockchain-Based Dynamic Key Management for Heterogeneous Intelligent Transportation Systems. IEEE Internet Things J. 4, 1832–1843. https://doi.org/10.1109/JIOT.2017.2740569

Li, B., Vorobeychik, Y., 2018. Evasion-Robust Classification on Binary Domains. ACM Trans. Knowl. Discov. Data 12, 1–32. https://doi.org/10.1145/3186282

Li, W., Wang, J., Fan, R., Zhang, Y., Guo, Q., Siddique, C., Ban, X. (Jeff), 2020. Short-term traffic state prediction from latent structures: Accuracy vs. efficiency. Transp. Res. Part C Emerg. Technol. 111, 72–90. https://doi.org/10.1016/j.trc.2019.12.007

Li, Y., Chen, Z., Yin, Y., Peeta, S., 2020. Deployment of roadside units to overcome connectivity gap in transportation networks with mixed traffic. Transp. Res. Part C Emerg. Technol. 111, 496–512. https://doi.org/10.1016/j.trc.2020.01.001

Liang, K.-Y., Mårtensson, J., Johansson, K.H., 2015. Heavy-duty vehicle platoon formation for fuel efficiency. IEEE Trans. Intell. Transp. Syst. 17, 1051–1061.

Linda, M., 2012. Car-hacking: Remote access and other security issues.

Liu, C., Li, B., Vorobeychik, Y., Oprea, A., 2017. Robust Linear Regression Against Training Data Poisoning, in: Proceedings of the 10th ACM Workshop on Artificial Intelligence and Security. Presented at the CCS '17: 2017 ACM SIGSAC Conference on Computer and Communications Security, ACM, Dallas Texas USA, pp. 91–102. https://doi.org/10.1145/3128572.3140447

Liu, F.T., Ting, K.M., Zhou, Z.-H., 2012. Isolation-Based Anomaly Detection. ACM Trans. Knowl. Discov. Data 6, 1–39. https://doi.org/10.1145/2133360.2133363

Liu, Y., Ning, P., Reiter, M.K., 2011. False Data Injection Attacks against State Estimation in Electric Power Grids. ACM Trans Inf Syst Secur 14. https://doi.org/10.1145/1952982.1952995

Luo, Y., Xiao, Y., Cheng, L., Peng, G., Yao, D. (Daphne), 2021. Deep Learning-based Anomaly Detection in Cyber-physical Systems: Progress and Opportunities. ACM Comput. Surv. 54, 1–36. https://doi.org/10.1145/3453155

Ma, S., Wen, F., Zhao, X., Wang, Z., Yang, D., 2019. An Efficient V2X Based Vehicle Localization Using Single RSU and Single Receiver. IEEE Access 7, 46114–46121. https://doi.org/10.1109/ACCESS.2019.2909796

Mannoni, V., Berg, V., Sesia, S., Perraud, E., 2019. A Comparison of the V2X Communication Systems: ITS-G5 and C-V2X, in: 2019 IEEE 89th Vehicular Technology Conference (VTC2019-Spring).

Presented at the 2019 IEEE 89th Vehicular Technology Conference (VTC2019-Spring), IEEE, Kuala Lumpur, Malaysia, pp. 1–5. https://doi.org/10.1109/VTCSpring.2019.8746562

MATLAB, D., 2021. Sensor Models - MATLAB & Simulink [WWW Document]. URL https://www.mathworks.com/help/nav/sensor-models.html (accessed 7.27.21).

Mei, S., Zhu, X., 2015. Using machine teaching to identify optimal training-set attacks on machine learners. Presented at the Twenty-Ninth AAAI Conference on Artificial Intelligence.

Milne, A., 2017. Hacking the railway.

Min, H., Wu, X., Cheng, C., Zhao, X., 2019. Kinematic and Dynamic Vehicle Model-Assisted Global Positioning Method for Autonomous Vehicles with Low-Cost GPS/Camera/In-Vehicle Sensors. Sensors 19, 5430. https://doi.org/10.3390/s19245430

Mollah, M.B., Zhao, J., Niyato, D., Guan, Y.L., Yuen, C., Sun, S., Lam, K.-Y., Koh, L.H., 2021. Blockchain for the Internet of Vehicles Towards Intelligent Transportation Systems: A Survey. IEEE Internet Things J. 8, 4157–4185. https://doi.org/10.1109/JIOT.2020.3028368

Mousavinejad, E., Yang, F., Han, Q.-L., Ge, X., Vlacic, L., 2020. Distributed Cyber Attacks Detection and Recovery Mechanism for Vehicle Platooning. IEEE Trans. Intell. Transp. Syst. 21, 3821–3834. https://doi.org/10.1109/TITS.2019.2934481

Nagao, W., Manabe, Y., Okamoto, T., 2005. A universally composable secure channel based on the KEM-DEM framework. Presented at the Theory of Cryptography Conference, Springer, pp. 426–444.

Nakov, S., 2022. Practical Cryptography for Developers.

Nie, T., Song, C., Zhi, X., 2010. Performance evaluation of DES and Blowfish algorithms. Presented at the 2010 International conference on biomedical engineering and computer science, IEEE, pp. 1–4.

Nie, T., Zhang, T., 2009. A study of DES and Blowfish encryption algorithm. Presented at the Tencon 2009-2009 IEEE Region 10 Conference, IEEE, pp. 1–4.

Nielson, S.J., Monson, C.K., 2019. Practical Cryptography in Python: Learning Correct Cryptography by Example. Apress.

Noureldin, A., Karamat, T.B., Eberts, M.D., El-Shafie, A., 2009. Performance Enhancement of MEMS-Based INS/GPS Integration for Low-Cost Navigation Applications. IEEE Trans. Veh. Technol. 58, 1077–1096. https://doi.org/10.1109/TVT.2008.926076

NovAtel, n.d. NovAtel SPAN on ProPak6 Datasheet [WWW Document]. URL https://novatel.com/ (accessed 7.30.21).

Papernot, N., McDaniel, P., Jha, S., Fredrikson, M., Celik, Z.B., Swami, A., 2016. The limitations of deep learning in adversarial settings. Presented at the 2016 IEEE European symposium on security and privacy (EuroS&P), IEEE, pp. 372–387.

Parkinson, S., Ward, P., Wilson, K., Miller, J., 2017. Cyber Threats Facing Autonomous and Connected Vehicles: Future Challenges. IEEE Trans. Intell. Transp. Syst. 18, 2898–2915. https://doi.org/10.1109/TITS.2017.2665968

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., Duchesnay, É., 2011. Scikit-learn: Machine Learning in Python. J. Mach. Learn. Res. 12, 2825–2830.

Pegues, J., 2017. Days before inauguration, hackers breached D.C. traffic and security cameras [WWW Document]. URL https://www.cbsnews.com/news/days-before-inauguration-hackers-breached-traffic-security-cameras-around-washington-dc/ (accessed 4.3.22).

Petit, J., Shladover, S.E., 2015. Potential Cyberattacks on Automated Vehicles. IEEE Trans. Intell. Transp. Syst. 16, 546–556. https://doi.org/10.1109/TITS.2014.2342271

Petit, J., Stottelaar, B., Feiri, M., Kargl, F., 2015. Remote attacks on automated vehicles sensors: Experiments on camera and lidar. Black Hat Eur. 11, 995.

Pous, N., Gingras, D., Gruyer, D., 2017. Intelligent Vehicle Embedded Sensors Fault Detection and Isolation Using Analytical Redundancy and Nonlinear Transformations. J. Control Sci. Eng. 2017, 1–10. https://doi.org/10.1155/2017/1763934

Psiaki, M.L., Humphreys, T.E., 2016. GNSS Spoofing and Detection. Proc. IEEE 104, 1258–1270. https://doi.org/10.1109/JPROC.2016.2526658

Reda, H.T., Anwar, A., Mahmood, A.N., Tari, Z., 2021. A Taxonomy of Cyber Defence Strategies Against False Data Attacks in Smart Grid. ArXiv210316085 Cs Eess.

Sanchez-Hernandez, C., Boyd, D.S., Foody, G.M., 2007. One-Class Classification for Mapping a Specific Land-Cover Class: SVDD Classification of Fenland. IEEE Trans. Geosci. Remote Sens. 45, 1061–1073. https://doi.org/10.1109/TGRS.2006.890414

Scaman, K., Virmaux, A., 2019. Lipschitz regularity of deep neural networks: analysis and efficient estimation.

Schmidt, D., Radke, K., Camtepe, S., Foo, E., Ren, M., 2016. A Survey and Analysis of the GNSS Spoofing Threat and Countermeasures. ACM Comput. Surv. 48, 64:1-64:31. https://doi.org/10.1145/2897166

Schreiber, M., Königshof, H., Hellmund, A.-M., Stiller, C., 2016. Vehicle localization with tightly coupled GNSS and visual odometry, in: 2016 IEEE Intelligent Vehicles Symposium (IV). pp. 858–863. https://doi.org/10.1109/IVS.2016.7535488

Shafaei, S., Kugele, S., Osman, M.H., Knoll, A., 2018. Uncertainty in Machine Learning: A Safety Perspective on Autonomous Driving, in: Gallina, B., Skavhaug, A., Schoitsch, E., Bitsch, F. (Eds.), Computer Safety, Reliability, and Security. Springer International Publishing, Cham, pp. 458–464.

Shafahi, A., Huang, W.R., Najibi, M., Suciu, O., Studer, C., Dumitras, T., Goldstein, T., 2018. Poison Frogs! Targeted Clean-Label Poisoning Attacks on Neural Networks, in: Advances in Neural Information Processing Systems. Curran Associates, Inc.

Shaheen, S., Chan, N., 2016. Mobility and the sharing economy: Potential to facilitate the first-and last-mile public transit connections. Built Environ. 42, 573–588.

Sheehan, B., Murphy, F., Mullins, M., Ryan, C., 2019. Connected and autonomous vehicles: A cyber-risk classification framework. Transp. Res. Part Policy Pract. 124, 523–536. https://doi.org/10.1016/j.tra.2018.06.033

Shen, J., Won, J.Y., Chen, Z., Chen, Q.A., 2020a. Drift with Devil: Security of Multi-Sensor Fusion based Localization in High-Level Autonomous Driving under GPS Spoofing (Extended Version). ArXiv200610318 Cs.

Shen, J., Won, J.Y., Chen, Z., Chen, Q.A., 2020b. Drift with Devil: Security of Multi-Sensor Fusion based Localization in High-Level Autonomous Driving under GPS Spoofing, in: 29th USENIX Security Symposium (USENIX Security 20). USENIX Association, pp. 931–948.

Shen, Z.-J.M., Feng, B., Mao, C., Ran, L., 2019. Optimization models for electric vehicle service operations: A literature review. Transp. Res. Part B Methodol. 128, 462–477.

Shimizu, T., Lu, H., Kenney, J., Nakamura, S., 2019. Comparison of DSRC and LTE-V2X PC5 mode 4 performance in high vehicle density scenarios. Presented at the Proceedings of the ITS World Congress, pp. 1–7.

Singh, P.K., Tripathi, P., Kumar, R., Kumar, D., 2017. Secure Data Transmission. Int. Res. J. Eng. Technol. 4, 217–222.

Skog, I., Handel, P., 2009. In-Car Positioning and Navigation Technologies—A Survey. IEEE Trans. Intell. Transp. Syst. 10, 4–21. https://doi.org/10.1109/TITS.2008.2011712

Smith, 2018. Akamai: DDoS attacks on the rise, become more specialized [WWW Document]. CSO Online. URL https://www.csoonline.com/article/3284349/ddos-attacks-on-the-rise-china-and-russia-behind-most-credential-abuse-attacks-report.html (accessed 4.3.22).

Sun, Z., Ban, X. (Jeff), 2013. Vehicle classification using GPS data. Transp. Res. Part C Emerg. Technol. 37, 102–117. https://doi.org/10.1016/j.trc.2013.09.015

Suya, F., Mahloujifar, S., Suri, A., Evans, D., Tian, Y., 2021. Model-Targeted Poisoning Attacks with Provable Convergence.

Toch, E., Lerner, B., Ben-Zion, E., Ben-Gal, I., 2019. Analyzing large-scale human mobility data: a survey of machine learning methods and applications. Knowl. Inf. Syst. 58, 501–523. https://doi.org/10.1007/s10115-018-1186-x

Toledo-Moreo, R., Zamora-Izquierdo, M.A., 2010. Collision avoidance support in roads with lateral and longitudinal maneuver prediction by fusing GPS/IMU and digital maps. Transp. Res. Part C Emerg. Technol. 18, 611–625. https://doi.org/10.1016/j.trc.2010.01.001

Tramèr, F., Boneh, D., 2021. Differentially Private Learning Needs Better Features (or Much More Data). https://doi.org/10.48550/arXiv.2011.11660

USDOT, 2020. USDOT ITS Joint Program Office's Strategic Plan 2020 - 2025.

van Wyk, F., Wang, Y., Khojandi, A., Masoud, N., 2020. Real-Time Sensor Anomaly Detection and Identification in Automated Vehicles. IEEE Trans. Intell. Transp. Syst. 21, 1264–1276. https://doi.org/10.1109/TITS.2019.2906038

Vorobeychik, Y., Kantarcioglu, M., 2018. Adversarial Machine Learning. Synth. Lect. Artif. Intell. Mach. Learn. 12, 1–169. https://doi.org/10.2200/S00861ED1V01Y201806AIM039

Wang, J., Shao, Y., Ge, Y., Yu, R., 2019. A survey of vehicle to everything (V2X) testing. Sensors 19, 334.

Wang, M., 2018. Infrastructure assisted adaptive driving to stabilise heterogeneous vehicle strings. Transp. Res. Part C Emerg. Technol. 91, 276–295. https://doi.org/10.1016/j.trc.2018.04.010

Wang, Yuanzhe, Liu, Q., Mihankhah, E., Lv, C., Wang, D., 2021a. Detection and Isolation of Sensor Attacks for Autonomous Vehicles: Framework, Algorithms, and Validation. IEEE Trans. Intell. Transp. Syst. 1–13. https://doi.org/10.1109/TITS.2021.3077015

Wang, Yuanzhe, Liu, Q., Mihankhah, E., Lv, C., Wang, D., 2021b. Detection and Isolation of Sensor Attacks for Autonomous Vehicles: Framework, Algorithms, and Validation. IEEE Trans. Intell. Transp. Syst. 1–13. https://doi.org/10.1109/TITS.2021.3077015

Wang, Yiyang, Masoud, N., Khojandi, A., 2021. Real-Time Sensor Anomaly Detection and Recovery in Connected Automated Vehicle Sensors. IEEE Trans. Intell. Transp. Syst. 22, 1411–1421. https://doi.org/10.1109/TITS.2020.2970295

Waze, 2021. Waze Beacon program [WWW Document]. URL https://www.waze.com/beacons (accessed 11.12.21).

Xiao, H., Biggio, B., Brown, G., Fumera, G., Eckert, C., Roli, F., 2015. Is Feature Selection Secure against Training Data Poisoning?, in: Proceedings of the 32nd International Conference on Machine Learning. Presented at the International Conference on Machine Learning, PMLR, pp. 1689–1698.

Xie, S., Wang, H., Kong, Y., Hong, Y., 2022a. Universal 3-Dimensional Perturbations for Black-Box Attacks on Video Recognition Systems. IEEE Secur. Priv. 19.

Xie, S., Wang, H., Kong, Y., Hong, Y., 2022b. Universal 3-Dimensional Perturbations for Black-Box Attacks on Video Recognition Systems. IEEE Secur. Priv. 19.

Yang, X., Sun, Z., Ban, X., Holguín-Veras, J., 2014. Urban Freight Delivery Stop Identification with GPS Data. Transp. Res. Rec. J. Transp. Res. Board 2411, 55–61. https://doi.org/10.3141/2411-07

Yu, J.J.Q., 2021. Sybil Attack Identification for Crowdsourced Navigation: A Self-Supervised Deep Learning Approach. IEEE Trans. Intell. Transp. Syst. 22, 4622–4634. https://doi.org/10.1109/TITS.2020.3036085

Yuan, Y., Wang, F.-Y., 2016. Towards blockchain-based intelligent transportation systems, in: 2016 IEEE 19th International Conference on Intelligent Transportation Systems (ITSC). Presented at the 2016 IEEE 19th International Conference on Intelligent Transportation Systems (ITSC), pp. 2663–2668. https://doi.org/10.1109/ITSC.2016.7795984

Zhang, F., Chan, P.P.K., Biggio, B., Yeung, D.S., Roli, F., 2016. Adversarial Feature Selection Against Evasion Attacks. IEEE Trans. Cybern. 46, 766–777. https://doi.org/10.1109/TCYB.2015.2415032

Zhang, Z., Zhang, W., Qin, Z., Hu, S., Qian, Z., Chen, X., 2021. A secure channel established by the PF-CL-AKA protocol with two-way ID-based authentication in advance for the 5G-based wireless mobile network. Presented at the 2021 IEEE Asia Conference on Information Engineering (ACIE), IEEE, pp. 11–15.