BEHAVIORAL MODEL BASED TRUST MANAGEMENT DESIGN FOR IOT AT SCALE

by

Brennan M Huber

Farah Kandah
Professor of Computer Science
(Chair)

Anthony Skjellum
Professor and Director, SimCenter
Computer Science and Engineering
(Committee Member)

Craig Tanis
Professor of Computer Science
(Committee Member)

Michael Ward
Professor of Computer Science
(Committee Member)

BEHAVIORAL MODEL BASED TRUST MANAGEMENT DESIGN FOR IOT AT SCALE

by

Brennan M Huber

A Thesis Submitted to the Faculty of the University of Tennessee at Chattanooga in Partial Fulfillment of the Requirements of the Degree of Master of Science: Computer Science

The University of Tennessee at Chattanooga
Chattanooga, Tennessee

August 2020

ABSTRACT

With the rise in the number of devices in the Internet of Things (IoT), the number of malicious devices will also drastically increase. Smart cities' decisions are based on data being collected by IoT devices in real-time, of which a connected-vehicle system is included. Behaviors such as malicious data injection can significantly impact connected vehicles. To aid in combating this threat, monitoring smart city and connected vehicle's sensor data will allow for construction of a behavioral model. Implementing machine learning will aid in constructing a standard behavior such that any device that begins to malfunction or behave maliciously can be detected and mitigated in real-time. This behavioral analysis will be further applied to supplement trust management approaches such that a more accurate value can be associated with the device's perceived trustworthiness without the need to rely on a majority consensus.

ACKNOWLEDGMENTS

First and foremost, I would like to thank my family and friends. To do this alone would be impossible for me and I truly am thankful for each and every one of you, and the support you all have given me, especially the one who was by my side the entire time.

I would also like to thank my co-workers for being incredibly understanding when I would have to leave work for classes, a random school meeting, or traveling across country for a conference. Without you, this would have been a much more stressful time, thank you.

I would like to also thank my peers who truly showed me how much enjoyment research and developing new projects could be, specifically Sai and Amani. The countless hours of meetings and the seemingly endless nights of working on BLAST were not only some of the times I learned the most, but those hours we spent working revealed my passion for research and how much fun it could be creating something brand new.

I cannot thank my professors and mentors enough. Dr. Skjellum, one of the most brilliant mentors is always able to provide so much useful and interesting information that I would leave our meetings with so many new ideas in my head that I would spend hours researching attempting to catch up.

Lastly, Dr. Farah Kandah! Little did I know that coming into your office to ask some questions about computer networks would lead to such a journey. The hours of conversations and countless last minute papers and proposals, I will never be able to thank you enough for the opportunities you have provided me.

Everyone I have mentioned is a part of what got me here and I am forever grateful. Thank you all.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

Introduction

The rapid growth of connected devices comprising the Internet of Things (IoT) is transforming traditional elements of city life into next-generation intelligent smart cities where decisions are based on data being collected by IoT devices in real-time. Among the key components contributing to smart cities' initiatives is the intelligent transportation system (ITS), which contains but is not limited to connected vehicles. The technology behind connected vehicles will enable vehicles to communicate with their peers, roadside units (RSUs), and other infrastructure to share vital transportation information such as current road conditions, congested traffic, and vehicular collisions [3, 4].

However, as the number of connected devices increases, the number of malicious devices in the system will also likely rise. The cybersecurity requirements of smart cities are distinct from conventional and past security issues, as they are constantly evolving because of new trends in technology and use cases [5]. The network setup can raise another challenge where frequent topology changes and high mobility characteristics of connected vehicles can create additional challenges in which cryptographic solutions cannot perform as well as expected so attackers can easily overtake authorized and authenticated users [6].

Because of the cybersecurity challenges that smart cities suffer from, it was necessary to create a new approach that sought to build trust between devices in an untrusted environment [6]. One solution offered is the design and implementation of trust management systems, in which devices will interact with one another and upon analysis of the data received, a device will learn to either trust or not to trust specific devices [7]. The decision to trust or to not trust is driven by

the other device' or devices' trust value(s), where a higher trust value indicates that the device has sent overall accurate messages while a lower trust could imply that the device is malfunctioning or is acting maliciously [7]. Trust management systems [7–10] enable devices to more quickly determine which other devices are sending accurate data and which are transmitting malicious or inaccurate data, based upon this trust value. The decision to increase or decrease another device's trust value is determined by using the device's own sensors to analyze the accuracy of the message, where the neighboring devices will then come to a consensus or agreement on the factual representation of the data. If the devices agree on the validity of the message then the initial device's trust value will be increased and conversely decreased if the message is deemed inaccurate. By enabling trust within smart cities, devices will be able to quickly accept data and execute decisions with a level of confidence that the data received is accurate.

As a result of the fundamentally untrusted environment that is a smart city and connected vehicles, it is difficult for vehicles to evaluate the credibility of received messages. Trust management systems have been shown to mitigate threats [6], however there are unique threats targeting the consensus mechanisms of trust management approaches such as colluding attacks [11], in which vehicles who behave maliciously will always not be removed from the system but instead rewarded for all coming to a majority consensus on the malicious data. Thus, relying solely on a majority consensus of devices is often not enough to mitigate threats, and therefore it is both critical and urgent to design and implement a solution that is capable of monitoring the data and behavior of such devices.

Through monitoring of connected vehicles' driving statistics, a localized behavioral model can be constructed that will accurately represent the standard behavior of vehicles driving in this area. This behavioral model will allow for future data to be compared to determine whether the data reasonably fits the expected behavior for that area. The results from this analysis of data will aid in the calculation of device behavior that can be further applied to more accurately represent the device's perceived trustworthiness in the system, that in its current state relies on a majority

consensus instead of the accuracy of the data itself. However, the need to remove threats against the majority consensus mechanism in current trust management approaches requires another method of determining data accuracy, providing motivation for machine learning to be implemented to construct this behavioral model. Given the numerous sensors and the massive amounts of data necessary for a connected vehicle network [12], machine learning will be capable of compiling the information to form the standard behavioral model that can be used as a basis for future data comparisons. This machine-learning approach allow for a more rapid detection and mitigation of threats which will further enable the real-time security needs of smart cities and connected vehicle networks.

1.1 Research Questions

In this thesis, we address the following research questions:

- Is it possible for a vehicle's behavior to be monitored and assessed in evaluating the trust in the network?
- What data is necessary to determine the overall behavior of a vehicle?
- How can machine learning enhance traditional trust management approaches?
- How will this behavioral analysis model assist in mitigating threats better than traditional trust management approaches?
- What level of accuracy is necessary for the behavioral analysis model to mitigate threats?
- What machine learning algorithm will best meet the real-time requirements of smart cities?

1.2 Motivation and Contributions

For trust management approaches to be successful, there is a critical and urgent need to detect and mitigate threats in real time. Emerging technologies such as smart cities and connected vehicles require novel cybersecurity approaches that are able to meet the needs that these advanced technologies require, trust management had proven its ability to satisfy these needs [7–10]. However, it is vital that trust management systems adapt to prevent new threats and detect and mitigate existing threats in real time. To accomplish this and to allow for a expeditious reaction to malicious behavior, the contributions of this work can be summarized as follows:

- **Construct of a behavioral model:** Constructing a behavioral model involves monitoring each vehicle's driving statistics to gathering a understanding of a specific geographical location and the typical behavior of the vehicles that participate there. The monitoring of driving statistics is vital to the real-time nature the system provides because to detect and mitigate threats in real-time, the system needs to have the most recent data available. Collecting data must be done for a small geographical area so that the system does not become impacted, which would negatively impact the real-time needs of the system. This behavioral model forms the foundation to the remainder of the thesis and can be expanded upon by creating many different behavioral models each corresponding to their own geographical area.

- **Develop a method to compare current behavior to the known behavioral model:** Using the behavioral model and the continued monitoring of vehicle driving statistics, the system will then be capable of creating a behavioral pattern identification process in which a vehicle's current behavior will be compared against the known behavioral model. Any vehicle that does not reasonably model the standard or expected behavior can be assumed to be an anomaly. These anomalies will provide the means of detection in this system, which will in turn lead to a mitigation process. Further, classification of the behavior is critical to the evaluation of these anomalies, because devices could be malicious/malfunctioning and thus classifying their behavior will aid in mitigation of such threats. The analysis accomplished will provide

critical insight into the detection and mitigation of threats in the system.

- **Formulate behavior-based trust mechanism** To further aid in the determination of the perceived trustworthiness of a vehicle, a behavior-based formula will be used to derive a distinct value that directly correlates to a given vehicle's likelihood of following the established pattern. This behavioral value can then be used in future implementations of trust management systems to more accurately define and calculate any vehicle's overall trustworthiness.

1.3 Organization

The remainder of the thesis is organized as follows. A brief discussion of key background information in Chapter 2 is followed by related work in Chapter 3. Chapter 4 presents the simulation and data collection approach. Methodology and experimentation are described in Chapter 5. Chapter 6 describes the evaluation and experimental results for the system. Finally, we conclude and discuss future directions in Chapter 7.

CHAPTER 2

Background Information

This chapter explains key ideas discussed throughout this thesis. The concepts covered here are the Internet of Things and Smart Cities in Section 2.1, Trust Management Systems in Section 2.2, Machine Learning in Section 2.3, and the threat model in Section 2.4.

2.1 IoT and Smart Cities

As technology progresses, it is becoming an integral part of every day life. Businesses, communities, and governments all rely on the internet to transmit information. Any device connected to the internet is a part of the Internet of Things. While they all have their individual purposes, the ability to transmit data across the internet is what unites them [13, 14]. There are, at present, an estimated 24 billion IoT devices [15]. IoT devices come in all shapes and sizes, such as a smart watch, smart refrigerator, security systems, smartphones, and even devices whose sole purpose is to collect and transmit data (such as thermometers, infrared detectors, motion detectors, or accelerometers).

2.1.1 Smart City

In [16], the following definition is offered: "A smart city is a framework, predominantly composed of Information and Communication Technologies (ICT), to develop, deploy and promote sustainable development practices to address growing urbanization challenges." A smart city

thus is a conglomeration of various devices that communicate to enable safer and more efficient operations [17]. Smart cities are composed of a new class of devices such as smart traffic lights, which have been shown to alleviate traffic [18]. Roadside Units (RSUs) are devices that are more computationally powerful than standard IoT devices; RSUs can collect the data from other devices and help coordinate the city such that it operates more efficiently [3, 4, 19]. Specific to this thesis, smart cities' initiatives include an intelligent transportation system (ITS), which is composed of connected vehicles (see Figure 2.1).

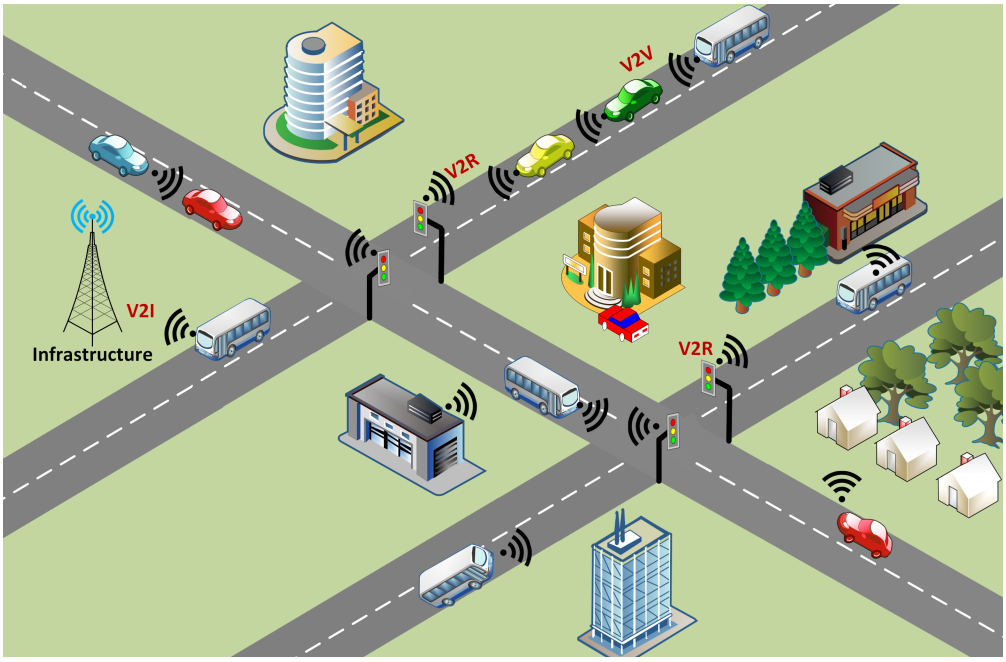

Figure 2.1 Smart city demonstrating how various devices are able to communicate through, vehicle-to-vehicle, vehicle-to-RSU, and vehicle-to-infrastructure to aid in efficient operations of the city

2.1.2 Connected Vehicles

Vehicle Ad Hoc Networks (VANETs) allow connected vehicles to communicate with other devices to share information regarding driving patterns or road conditions [3, 4].

To create a safer driving environment connected vehicles will need vast amounts of data in order to make efficient and safe decisions for it's passengers as well as members of the city. To accomplish this connected vehicles will need methods of communicating with the infrastructure that is dedicated to the operations of VANET-ITS. These communication protocols include vehicle-to-vehicle (V2V) enabling vehicles to communicate with one another and vehicle-to-everything (V2X), which allow the vehicles to communicate with all device's in the smart city such as RSUs [20].

The National Highway Traffic Safety Administration (NHTSA) describes vehicle-to-vehicle (V2V) communication as "Information between nearby vehicle to potentially warn drivers about dangerous situations that could lead to a crash" [12]. While connected vehicles are not currently popular with car manufacturers, it is estimated that to transform a standard vehicle into a connected vehicle would cost approximately $350 [12] to equip the vehicle with the necessary communication devices and sensors. Some of these sensors include ultrasonic sensors, cameras, and radars, which when used together should provide more information about the surrounding area than what a driver is capable of seeing of their own.

2.2 Trust Management Systems

New technologies such as smart cities and connected vehicles often require novel methods of cybersecurity to protect the users from malicious acts. One solution to address the cybersecurity needs of connected vehicles is through the application of a trust management system [7–9]. Trust management systems are a method which rewards or punishes devices based on how they operate in the network. Each device has a trust value that corresponds to the observed trustworthiness based

on historical interactions with other devices in the network. Devices interact and will evaluate the accuracy of the data each receives. If the data is determined to be accurate, then the other device will be rewarded with a higher trust value, if the data is inaccurate or malicious their trust value will be decreased [7]. This method of a group evaluating messages for accuracy is known as a majority consensus [21]. This majority consensus comes with an attendant problem: colluding attacks. This is when a group of devices all act maliciously and will agree on the accuracy of the data thus boosting their trust values [22].

As previously stated, smart cities require decisions to be made in real time (which means in our context sufficiently fast that decision-making can improve the overall operation of the smart city). But because of the untrusted nature of smart cities, devices cannot blindly accept data as such data has the potential for negative side effects. Trust management offers the ability to only accept data that comes from devices with a high trust factor so that data received is most likely accurate (however a high trust factor is not guarantee that the data is 100% accurate even if the other device has a perfect trust factor). This is a because history is unable to predict the future; any device has the capability of turning malicious, making it even more vital to develop a modernized solution.

2.3 Machine Learning

Machine learning has risen in popularity in recent years, especially with the growing interest in data analysis. Merriam Webster defines machine learning as "the process by which a computer is able to improve its own performance by continuously incorporating new data into an existing statistical model" [23]. This means that, through a mathematical model, a computer is able to be trained to critically analyze data such that it can make predictions of future outputs or provide a more concise explanation of the outputs without having to look through mass amounts of data. There are two types of machine learning: supervised and unsupervised [24]. Supervised

machine learning is where the developer will create the training set and manually define the data such that the machine learning algorithm will be able to identify patterns based upon these. On the other hand, unsupervised machine learning involves algorithms where the computer will groups the data together automatically [25].

There are many different applications for machine learning, some of the more popular include: speech recognition, computer vision, pattern recognition [26].

Machine learning terminology used throughout this thesis is as follows:

- True Positive and False Positive: A true positive is where a model correctly identifies an outcome, while false positives occur when a model predicts an outcome which does not actually occur.
- Training/Fitting: In order for machine learning to be able to make prediction, it must learn the data that it will be operating on. This learning process is referred to as training or fitting the model.
- Inlier and Outlier: After the machine learning algorithm has been trained certain algorithms are designed to determine if any new data that is input in the system matches the data that the algorithm was trained on. Data which does not match is an outlier, and conversely data which matches is an inlier.

2.3.1 Anomaly Detection

One specific class of machine learning is known as anomaly detection, which is common utilized, for instance, in cybersecurity [27, 28]. Like other types of machine learning, anomaly detection requires training on "standard" data. The algorithm will learn patterns and gain a baseline understanding of the data [29]. With the algorithm understanding what the standard behavior is, anything that differs sufficiently will be flagged as an anomaly (which can lead to a number of different mitigation techniques such as having a human take a look to determine what would've

caused such an event). Figure 2.2 demonstrates how different anomaly detection models group data points and shows that any data point not inside that grouping is considered to be an outlier (that is, an anomaly).

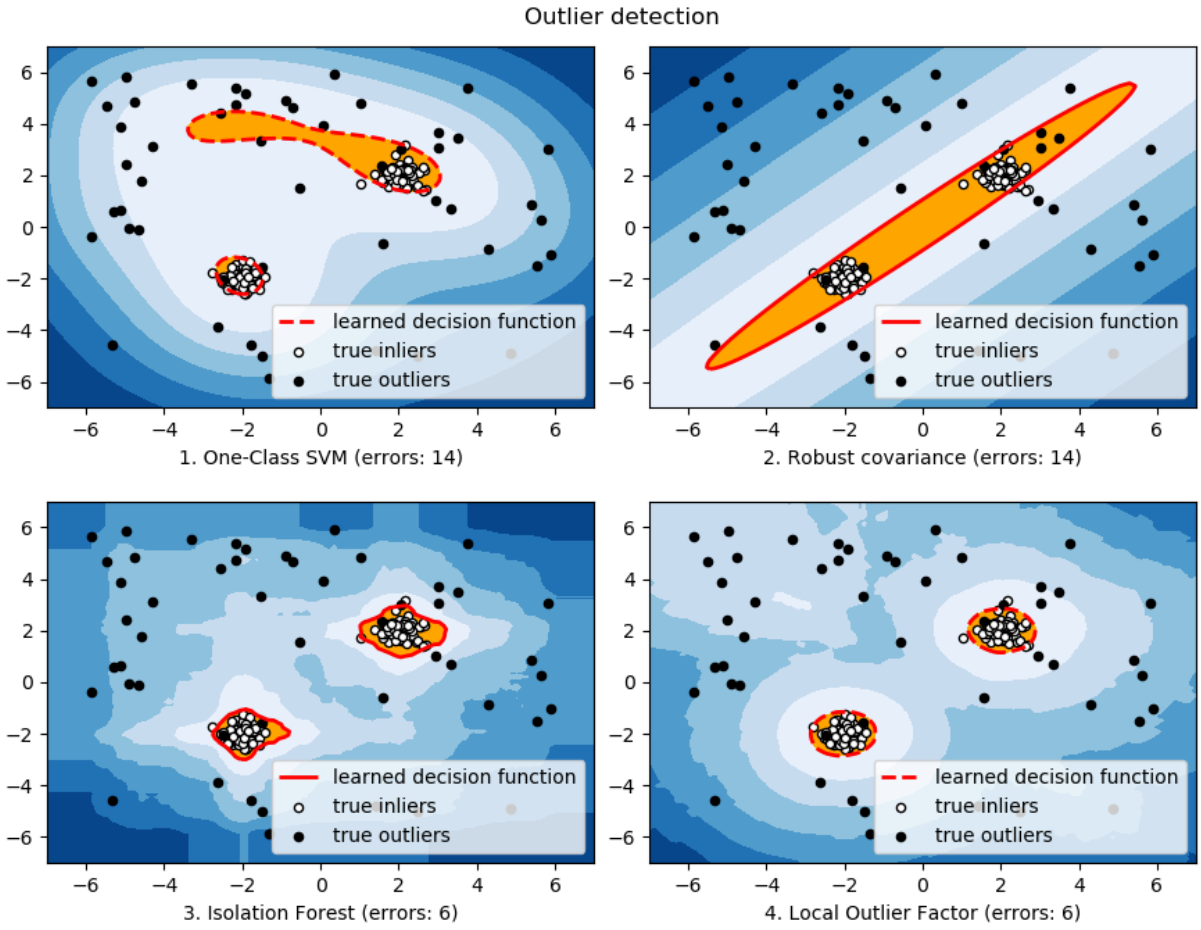

Figure 2.2 Visual representation of how anomaly detection algorithms (One-class SVM, Robust Covariance, Isolation forest, and LOF) determine if data is an inlier or outlier (adopted from [1])

2.3.2 Classification

Another area of machine learning is called either classifiers or classification algorithms. These are generally supervised models, where the algorithm will be trained on datasets that have a flag that corresponds to the specific class of data that it represents. Upon training, any new data that is input into the machine-learning algorithm will be labeled with the class to which it belongs [30]. Classification algorithms accomplish this through the grouping of data; this is necessary because it provides a distinction between different subsets of data such that new data can be classified according to its relation to the subset [30], a visual representation of this can be seen in Figure 2.4. An example of a classifier is image recognition software. If a machine learning algorithm were to be trained on images, in which each image was labeled corresponding to the animal that was pictured, then if you were to input another picture of an animal, then the machine learning algorithm would recognize the image and output the type of animal.

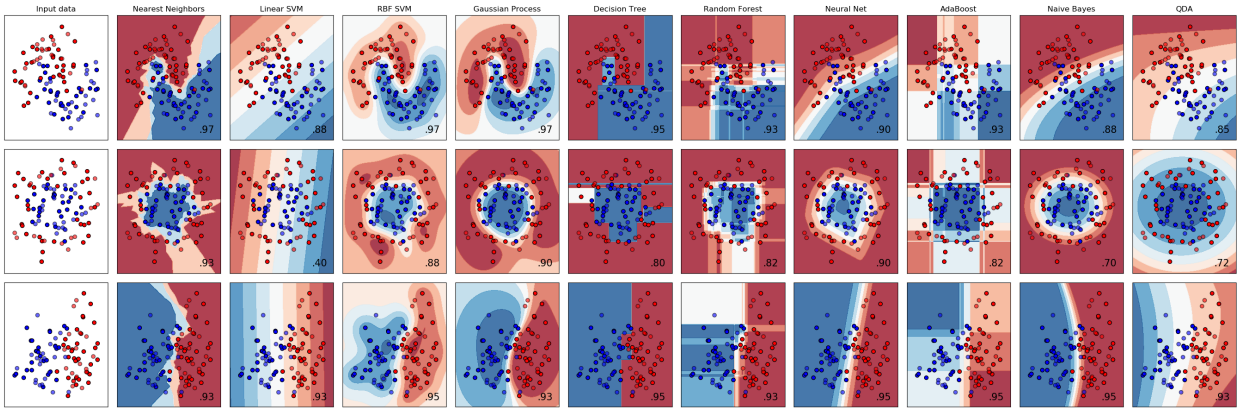

Figure 2.3 Visual representation of how classification machine learning algorithms group data together such that any point that is mapped to a specific location is classified based on that grouping (adopted from [2])

2.4 Threat Model

A number of malicious attacks affect the way IoT entities communicate with each other and could have an impact of the trust building in a network. This can affect the decision-making, which could lead to negatively effecting human lives. In this section, the threat model is presented, which comprise of a number of attacks with different targeted effect on the system, including breakout fraud, illusion-based attacks, and colluding. In the future, we will look into other threat models that will affect IoT enabled systems. The purpose of a behavioral trust monitoring is to aid in prevention of attacks that can occur in the system. Thus, by implementing this approach the following attacks can have minimal effect.

2.4.1 Breakout Fraud

Breakout fraud attack occurs when a device attempt to first build trust in the network through providing accurate information, but, at a given point in time, said device will begin acting maliciously by transmitting false data to other devices. This attack aims to take advantage of the fact that the device has earned a high trust value, implying that other devices in the network will most likely accept its data. At such a juncture, the malicious device will be able to successfully inject malicious data into the system, which can negatively impact the operations of the network.

2.4.2 Selective Behavior

Because of to the highly dynamic nature, connected vehicles can act maliciously with a high possibility that they randomly switch from sending accurate messages to inaccurate messages, and back again. Selective behavior attacks seek to maintain a high trust value such that, upon injecting malicious data, it is more likely for the data to be accepted (such that other devices will apply this data for a decision-making process). However, upon evaluating these messages, the vehicle's trust

will be lowered. Thus, the vehicle will opt to switch back to sending accurate messages for a short period of time. This variety of attack could be launched through randomly sending true and false messages (mixed behavior), or it can be done on a periodic schedule (flip-flopping).

2.4.3 Illusion-based Attacks

Illusion-based attacks represent a threat that occurs when vehicles have knowledge of the data that relates to critical messages in order to force their data to model those critical messages. This means that the malicious vehicle creates the illusion of a critical incident, even though the incident did not occur. This attack is especially dangerous since it has the potential to deceive protocols to prevent the injection of malicious data. By creating an illusion-based attack and circumventing protection protocols, the smart city would accept the data and be inclined to make decisions based on this new information. But, because this information is actually inaccurate, the smart city or connected-vehicle network could potentially make decisions that would negatively impact the overall condition of the network.

2.4.4 Colluding Attacks

In traditional trust management implementations using a majority-based consensus mechanism [11], there is the potential for a colluding attack or an overrule of the majority attack. This occurs when the any number of malicious devices reach consensus and the summation of those devices' trust values outweigh the actual trustworthy devices (such that the malicious information is accepted as accurate). One example is as follows: one device has a 100% trust value while another four malicious devices each have 26% trust. If the four vehicle with lower trust agreem, then the weighted trust values of those is higher than the trust value of the one telling the truth; thus, the consensus has now been compromised. This is an serious problem with traditional trust man-

agement approaches, smart cities and connected vehicles rely on the system being able to make real-time decisions when necessary. By not detecting the presence of malicious devices in the network, the malicious device or devices will have artificially inflated trust values, which can be used to propagate their malicious data throughout the network. Thus, causing the system to be unable to trust device(s) with high trust this attack has effectively shut down trust management system(s).

2.5 Concept of Operations

In a smart city with VANET-ITS [3, 4], such as connected vehicles, there must be a certain infrastructure setup which aids in the communication of vehicles. This will be divided into two parts: vehicles and stationary nodes (RSUs). Since these smart and connected vehicles are driving in the city, they will be using their sensors to detect information regarding current road conditions, this information will be transmitted to the RSUs (such that critical information can be extracted and used by the smart city [19]).

First, these vehicles will need to be equipped with sensors such that the vehicles are able to detect information pertaining to the road conditions. As stated above, these sensors can be devices such as ultrasonic sensors, cameras, or radars, which, when used together, will collaborate to provide information such that other vehicles on the road can make informed decisions [12]. Further, the vehicles must be equipped with an On-board Unit (OBU) that is used to enable for communication between the vehicle, other vehicles, and the smart-city infrastructure. These OBUs will be composed of Directed Short Range Distance (DSRC) technology that has been shown to satisfy vehicle-to-vehicle communication requirements such as rate of transfer and distances between devices [31].

There also need to be stationary nodes forming a collection of Road Side Units (RSUs), which, as stated previously, are more computationally powerful. This means that most of the data processing and analysis will be conducted on these devices [3,4]. RSUs will be located throughout

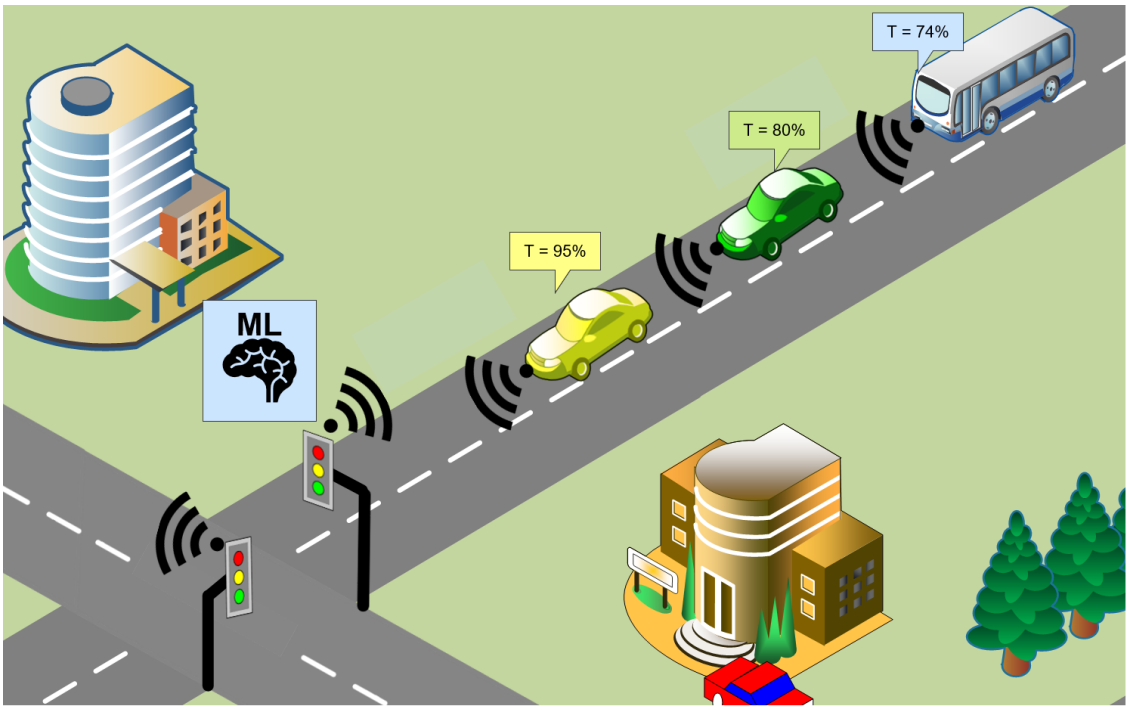

Figure 2.4 In this figure the traffic light is the RSU, the vehicles will be reporting their observations and data to this RSU as it is the closest in proximity. Each vehicle has its own trust value available such that informed decisions can be made based upon the trustworthiness of the data

the city such that each RSU is able to monitor a small area and when able to inter-communicate to complete coverage of the entire smart city [19]. Having multiple RSUs has been shown to reduce delay of messages and to increase the delivery ratio, while simultaneously reducing the number of retransmissions that vehicles must conduct to ensure that the messages is received when compared to a system with none or just one RSU [19].

CHAPTER 3

Related Work

3.1 Trust Management

In our context, trust is based on the history of interactions and the validity of the information exchanged between network entities [7–9]. Managing trust in the network has received significant attention since it can provide a dynamic layer of security where devices in the network will build a bond based on their interactions, ensuring that data can be transmitted in the network and accepted with confidence that it came from a trustworthy device [9, 10, 32]. Several trust management schemes have been proposed, including entity-based where trust is based on the device itself, data-based is when the trust is based upon the data that is being sent, and hybrid trust, which is a combination of both the authenticity of the device and the information being sent [6, 33].

One area of interest in cyber-physical systems is connected vehicles [33–37]. Compared to static networks, the dynamic nature of connected vehicles requires a distributed system that enables vehicles to gather and share information toward building trust in the network as they move from one place to another (this trust building can be achieved through collaboration between the connected vehicles and fixed roadside units) [38].

The work presented in [32] discussed the idea of using a "lead" vehicle acts as the primary source of truth for the network. In this work, the lead vehicles are only emergency response vehicles that are given a 100% trust value. In trust-based solutions, the lead vehicle is as vulnerable to outside attacks as any other vehicle in the system and thus should not be blindly trusted. Therefore, a way to have a distributed consensus to verify the data collected, and corresponding trust values, is required for the system to operate in a secure manner.

Kerr et al. provided an analysis of unique trust models and demonstrate how each deal with an adversarial model showing how the trust management system is capable of dealing with attacks.The threats provided in this work include injected messages where malicious vehicles would purposefully inject bad data into the network and Denial of Service (DoS) attacks that would flood the vehicles with data such that the vehicles would be unable to process all the information.

Previous work has proposed solutions for trust management implementation in Vehicle Ad Hoc Network-Intelligent Transportation Systems (VANET-ITS) (e.g., [33–37]). Yang et al. proposed a decentralized system, claiming that a centralized system is impractical for the growth that a VANET-ITS would require [33]. Another drawback of a centralized system is the massive overhead that could be caused if several vehicles should be communicating with the central node at once. By having several roadside units (RSUs) located throughout a city, each area within it can be divided appropriately (e.g., load balanced and/or geographically). Therefore, the processing load will be reasonably balanced. Further, the authors continued by proposing trust-factor calculations where each vehicle begins with a neutral value, and, as messages are passed between vehicles the trust value will be increased or decremented based on the accuracy of messages. The method for evaluating the accuracy of a message is based on the experiences that other vehicles in the network have had with a given message. The critical drawback of this approach is the scenario in which there are several malicious vehicles in the network and these vehicles collude to evaluate their messages as accurate. This scenario increases the malicious vehicles' trust factor, decreasing the overall integrity of the system.

3.1.1 Human-based Trust

Another study shows how humans evaluate a situational form of trust [39], where trust is based on the current environment that the person is experiencing. The study examines how an individual's trust will change in what is called global virtual teams (where team members are

not colocated) in the information systems field. It shows that a person's initial trustworthiness or perception of trustworthiness of the team members has a greater impact in the formation of trust. With this, it was generally found that a higher trust between team members led to more frequent communications (as it assures everyone is completing the necessary tasks) [39]. This study shows that if a team member does not initially prove themselves, then the overall trust of the team member is affected, and the team member struggles to gain trust later.

3.2 Behavior Analysis

Another study conducted in an IoT environment has demonstrated that a behavioral analysis of IoT devices can be implemented even with limited resources [40]. To accomplish this, the authors implemented a lead node that monitors the network traffic metadata that devices send. Data such as the source IP address, the destination IP address, the MAC address, and the port number, etc are extracted as features where the lead node then stores the data on a behavior monitor blockchain. Using this data and associated features a machine learning model was built to analyze the authenticity of the message. While the authenticity is important, this would only prevent attacks which an outside actor attempts to harm the system and would not mitigate threats from actors inside the network.

3.3 Machine Learning

Machine learning algorithms have been shown to aid in the cybersecurity requirements of smart cities [41–43]. Alrashdi et al. describe the challenges of implementing machine learning algorithms within smart cities such as limited resources of devices or the heterogeneous nature of IoT (which will lead to a higher false positive ratio resulting from differences in devices). Notwithstanding heterogeneity, their work shows that the implementation of machine learning can enhance

smart cities intrusion detection systems (IDS) for certain threats such as DoS, Worms or malicious programs installed to computers, Backdoors (which are vulnerabilities in specific systems that allow for unauthorized access), and so on. Garcia et al. state that, in traditional systems, authentication and confidentiality are satisfied by cryptographic solutions. However, because of the dynamic nature of wireless networks and smart cities, cryptographic solutions are insufficient because an attacker could capture a node, effectively bypassing such measures. This threat has led to the study of implementing machine learning for prevents attacks on smart cities.

3.3.1 Anomaly Detection

One specific class of machine learning is used to achieve anomaly detection; a common application for anomaly detection is in cybersecurity [27,28]. Anomaly detection requires training on "standard" data, where the model will then learn patterns and have a baseline understanding of the data [29].

Garcia et al. implemented different anomaly detection models, supervised and unsupervised, such as local outlier factor and support vector machines to see which would lead to better detection of threats [42]. Specifically, the authors implemented Mahalanobis Distance (MD), Local Outlier Factor (LOF), Hierarchical Clustering, and Support Vector Machines (SVM), which are all anomaly detection algorithms that will group the expected data together such that any data point which does map to the expected data is determined to be an anomaly. In particular, MD is a measurement of the distance between two points such that it is able to measure a new points distance from the mean of the original data set [44]. LOF is another measured distance-based algorithm however the numerical scale for determining outlierness is adjustable [45]. Hierarchical Clustering and SVMs both work by means of plotting data points and then clustering those data points [46, 47]. For anomaly detection purposes, if any new point is mapped outside of the cluster then the data is considered an anomaly. Through the implementations of different machine

learning algorithms, the team concluded that the SVM implementation was best suited for the data and accuracy which they were pursuing. However, the authors were limited in their results since they worried about the overhead inherent with a high number of false positives; to combat this possibility, they accepted a lower accuracy in their detection algorithm.

Another study presents a two-tier anomaly detection approach [43]. The first tier of machine learning in this aspect was dimensional reduction, which would then feed into the classification algorithm (Naive Bayes). The output of this first tier would be the initial stages of anomaly detection, which has the second tier further classifying data determined not to be an anomaly. This second tier's purpose is to enable the system to make more intelligent decisions based upon the data through the use of a K Nearest Neighbors algorithm. While this two-tiered approach is able to process data more efficiently than a single-tiered approach, its primary purpose is to offload the classification of standard or normal data to another classifier, which imposes less computational demand on the first machine learning algorithm.

3.3.2 Classification

Another area of machine learning is called classifiers or classification algorithms. These are generally supervised models, where the algorithm will be trained on datasets that have a flag that corresponds to the specific class of data that it represents. Upon training, any new data that is input into the machine learning algorithm will be labeled with the class of which it belongs to [30].

Other studies have been conducted that implement machine learning classification of smart city data, Chin et al. compiled weather sensor data from a smart city and attempted to classify the data such that weather predictions could be made [48]. Through testing of Naive Bayesian, J48 Tree, and Nearest Neighbor Classifiers, the authors were able to predict 100% of the cases tested: rainfall and temperature comparisons, thus showing that classifiers have a positive effect when it comes to analyze data to make better predictions in a smart city.

Brisimi et al. implemented machine learning algorithms such as SVMs, logistic regression, adaboost, and random forests to detect objects in the road [49]. Using an anomaly detection algorithm first to determine if there was a bump in the road, it would then feed into the classification algorithm to determine if the bump was from a pot hole, a road patch, or a sunk casting. With an accuracy of 86% through 88% the classification of bumps successfully enabled city officials to prioritize which roads need fixing.

3.4 Observations

Based upon the literature review, traditional trust management systems are only concerned with the trust values. This is often not enough because of how heavily these approaches rely on a consensus of the data giving way to threats that directly target trust management implementations such as breakout fraud and colluding attacks. Trust management implementations also do not meet the real-time needs that smart cities require. It often takes several instances of malicious behavior before the system can safely and effectively punish or remove devices that suddenly begin acting maliciously. Thus, it is vital to find a new implementation that can better prevent threats to smart cities and connected vehicles, while also maintaining the ability to process and punish malicious actors in real-time.

Machine learning has been shown to accurately detect and predict outcomes based on sensor data from devices within the smart city. Garcia et al. showed that anomaly detection was able to aid in the detection of threats in the system, but because of to the overhead associated with false positives, the authors accepted a lower accuracy rating. Pajouh et al. implemented a two-tier machine learning approach, which was able to detect anomalies while also using a classification algorithm to aid with the overhead of false positives.

Thus, it is both critical and urgent to design, develop, and implement a trust management approach that is capable of monitoring connected vehicles driving statistics to detect threats in

real-time. While simultaneously being able to mitigate these threats based upon the classification that is detected.

CHAPTER 4

Simulation

In this chapter, the sources of the data collection and generation process from this thesis will be discussed. 4.1 discusses vehicle-to-vehicle communications. Section 4.2 covers SUMO, the program used to simulate vehicle driving statistics. Messages generated are described in Section 4.3. The chapter concludes with a summary in Section 4.4.

4.1 V2V communications

The National Highway Traffic Safety Administration (NHTSA) describes vehicle-to-vehicle (V2V) communication messages as basic safety messages (BSMs) that correspond to messages passed between vehicles regarding dynamic information such as headings, speed, and location [12]. Further, there are other safety applications that depend on the sensors with which connected vehicles are equipped. Potential messages or warnings include the following (as defined in [12]):

- **Intersection Movement Assist (IMA):** IMA alerts drivers when it is unsafe to merge into an intersection.
- **Left Turn Assist (LTA):** LTA warns the driver that it is unsafe to make a left turn as there is oncoming traffic and thus a potential for collision.
- **Emergency Electronic Brake Light:** A device that alerts other vehicles when a driver is applying the brakes. This is useful when the front vehicle might not be visible to the follower vehicle because of a blind curve or severe weather conditions.

- **Forward Collision Warning:** Warns the driver of potential collision with the leading vehicle. Such a warning would also be beneficial to other drivers and infrastructure because it is a measurement of how close vehicles are following.
- **Do-not-pass warning:** Communication warning following vehicle that it is not safe to pass. This is attributed to a number of reasons (such as oncoming traffic is approaching and so it is not safe to pass at present).

If every vehicle had the necessary sensors and the ability to produce warnings messages, the NHTSA predicts that IMA and LTA will prevent between 400,000 and 600,000 crashes; 190,000 to 270,000 injuries; and save close to 1,000 lives each year. These technologies could prevent nearly 80% of all non-alcohol related incidents [12], making it imperative to protect the data that is being transmitted between these vehicles.

With the potential benefits that come from V2V communication, it is crucial to ensure the security of the V2V messages (since without it any malicious actor can impersonate or eavesdrop on the communications between vehicles) [50]. Using asymmetric encryption through a public key infrastructure has been shown to be an effective means for the security of V2V communications [50, 51]. However, the time requirements did not allow for real-time processing of messages when the number of vehicles increased. This thesis seeks to add a dynamic layer of security that better meets the real-time needs of connected vehicle so the study of the encryption, authenticity, and integrity of V2V messages are out of scope.

4.2 Simulation of Urban Mobility

Because connected vehicles being a relatively new research area, and few exist in production environments, real-world data is not readily available enough for the demands of this thesis. Thus through the usage of the Simulation of Urban MObility (SUMO) it is possible to accurately simulate large scale road networks with an abundance of cars where at each time step data is logged

and output [52, 53]. This output can be parsed to extract features that is described in more detail in subsection 4.3.

SUMO is an open source program developed by a German Aerospace Center in 2001 as a basic traffic simulation package [53]. Since then SUMO has become feature packed with many different packages that aid in traffic simulation.

SUMO allows for the ability to design your own custom maps with the `neteditor` tool and accept many different sources as input such as a map obtained from OpenStreetMap (OSM) [53]. OSM is a user generated street map that is capable of being exported in a way that maintains the data such as street names and other geographical features. SUMO includes tools that are able to convert this OSM exported map into a format that SUMO is able to process [54]. This allows users to be able to load custom maps of cities from all around the world.

SUMO is also capable of handling different routing protocols and generating vehicles on demand, for the desired map, that allows for the possibility of creating numerous unique simulations that can accurately represent real-world traffic patterns. SUMO also comes with a suite of tools which includes one capable of generating a random number of vehicles with completely random routes allowing for dynamic navigation patterns [52, 53]. This provides the ability to obtain unique information for each simulation, providing a wider range of data.

The simulation aspect of SUMO is microscopic meaning that the simulation tracks each individual vehicle by a unique identifier and each vehicle's route is described in a configuration file [53]. This allows for the potential to completely fine tune the simulation such that vehicles will travel the path that the developer creates. SUMO is also described as a time-discrete simulation that means that it is capable of manually defining the time step at intervals ranging from 1 second to 1 millisecond [52]. This time setting will determine the step length of outputs such that one simulation can be as coarse or granular as desired.

SUMO is comprised of a command line tool as well as a Graphical User Interface (GUI), while the command line tool is able to run and output the exact same as the GUI, the GUI offers

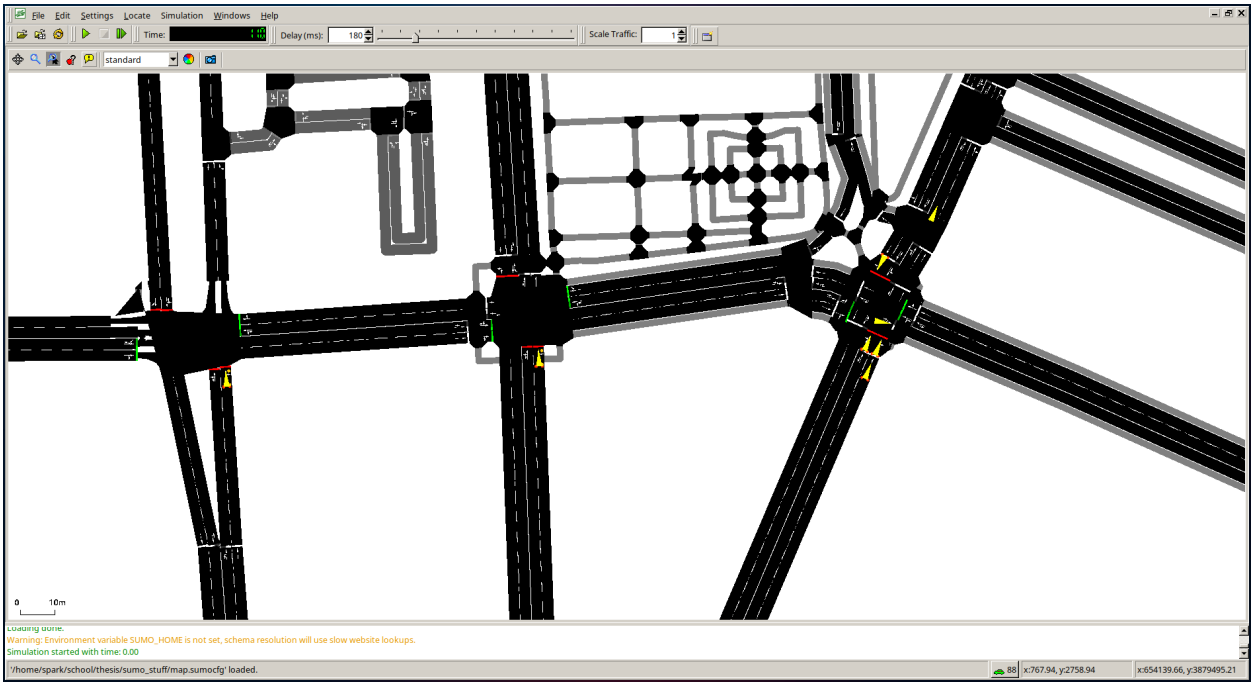

Figure 4.1 Example showing Chattanooga, TN Market street visualized in a SUMO's GUI simulation. Vehicles are represented by the yellow triangles, as well as traffic lights being visible at each intersection

a better user experience through customized visuals and the ability to interact with scenarios like traffic lights and re-routing of vehicles [52]. When a new simulation begins vehicles will be staggered into the system, one entering at each new time step. Each vehicle will have it's route defined in the configuration file and the vehicle will continue on this route until it exits the map. When all vehicles leave the system the program will end and the data will be saved into its corresponding `XML` files, where `XML` is an extensible markup language that is organized in a tree structure such that information is related hierarchically [55].

Based upon different arguments that can be applied to the simulation, SUMO is capable of producing numerous outputs [56]. Such outputs include information such as the position and speed of all the vehicles, as well as emission values, trajectory data of the vehicles, and surrogate safety measures which is information directly related to safety measurements such as braking rates. There also exist lane values that correspond to the edges or lanes in the simulation, it is capable of outputting emission measurements for that lane, the noise level of the road, even how many vehicles were on the road at a particular instance.

4.3 Message Generation

According to the NHTSA recommendations, SUMO is capable of generating reports including the numerous features such as speed, position, acceleration, braking, etc. These features can also as seen in Figure 4.2.

Between NHTSA's information regarding V2V communication and SUMOs simulation and data capabilities, these features will be used to monitor the driving statistics of each vehicle in the simulation. In addition to these features, the data will also contain a rank or message that correlates the driving statistics to warning (or basic safety messages) and incident messages. Table 4.1 shows the correlation of message ranks to a particular V2V communication message.

For the purposes of this thesis, these V2V communications will be manually defined. A

timestep_time	vehicle_id	vehicle_pos	vehicle_speed	vehicle_x	vehicle_y	motionState_acceleration	motionState_speed	brake_rate	Message
0	0	5.1	0	1948.94	2807.72	0	0	0	1
1	0	7.24	2.14	1950.9	2806.86	2143	214	0	3
1	1	5.1	0	2307.63	3033.84	0	0	0	1
2	2	5.1	0	897.83	1569.26	0	0	0	1
2	0	10.79	3.55	1954.16	2805.44	1407	355	0	1
2	1	7.62	2.52	2309.96	3032.87	2522	252	0	3
3	2	6.4	1.3	896.64	1569.78	1305	130	0	1
3	0	15.88	5.09	1958.81	2803.4	1536	509	0	3
3	1	11.77	4.15	2315.06	3034.34	1628	415	0	3
3	3	5.1	0	2126.91	2445.02	0	0	0	1
4	1	18.14	6.37	2320.95	3031.91	2216	637	0	3
4	0	22.59	6.71	1964.96	2800.71	1625	671	0	3
4	4	5.1	0	3070.18	1845.72	0	0	0	1
4	3	6.65	1.55	2126.29	2443.6	1550	155	0	3
4	2	9.37	2.96	893.93	1570.98	1658	296	0	3
5	5	5.1	0	4483.44	3839.77	0	0	0	1
5	3	10.73	4.08	2124.65	2439.86	2527	408	0	3
5	1	27.02	8.88	2329.17	3028.54	2514	888	0	3
5	0	31.25	8.66	1972.89	2797.23	1945	866	0	3
5	4	6.83	1.73	3071.76	1845.01	1728	173	0	3
5	2	14.3	4.93	889.42	1572.97	1968	493	0	3

Figure 4.2 SUMO simulation data snapshot showing the simulation time (sec), the vehicle ID, the vehicle's position, speed, location (x,y), acceleration, speed, brake rate and the message being reported by that vehicle

message rank 0 does not correspond to any message in particular. This form of message does not have any impact into the system but is implemented such that the vehicle is able to transmit its current driving statistics to the infrastructure so that the smart city is aware of the current statistics

Rank	**V2V Message**
0	No Message - No information related data available.
1	General Alert - Corresponding to specific sensor data such as rain detection.
2	Object on shoulder - Alerts others of an object detected on the shoulder of the road.
3	Change in driving patterns - A vehicle alerting others of an upcoming change in driving patterns.
4	Object in road - Incident message informing others of an object detected on the road.
5	Emergency vehicle - Message regarding the detection of an approaching emergency vehicle.
6	Car Collision - Incident message alerting others of a car collision.

Table 4.1 Communication messages are categorized into ranks based upon the severity of the information present in the message. 7 ranks are presented, starting from rank 0 being a no information to be shared, all the way to rank 6 providing critical information alerting others of the detection of a car crash

of drivers on the road.

Warning messages (ranks 1-3) will represent NHTSA's basic safety messages. These will be messages that should not warrant any drastic changes in driving patterns, but more so for the other drivers and infrastructure to be made aware of the current environment. In real-world applications, these messages could also be derived from another sensor. Specifically, rain detectors, a light detection for headlights, detection of ice, etc could be used as Message 1 or general alerts. A message with rank 2 might originate from an ultrasonic sensor or camera that is able to detect and identify objects on the shoulder of the road such as a traffic sign or a pedestrian [57]. The last warning message would be a message of rank 3. This is an alert to inform other drivers that there is a change in driving patterns. This message could, for instance, come from a sensor monitoring the acceleration of a vehicle or other information such as the blinker indicating a lane change.

There also exist incident messages (ranks 4-6), where it is expected to see a substantial change in the current driving pattern to accommodate the reason for the message. These incident messages correspond to ranks 4-6. An example of a message rank 4 is an object detected in the road, his could be a pot hole or an animal in the road. While a message rank 5 would correspond to an emergency vehicle being detected, in some ITS initiatives there is prioritization of emergency vehicles [58] thus it is critical to have the connected vehicles be able to acknowledge and prepare for this type of event. The last message of rank 6 would be for car collisions. Car accidents have the potential to seriously impact smart city ITS's, thus it is necessary to report accidents to the smart city infrastructure such that intelligent routing systems have the capability to reroute traffic when necessary [38]. The distinction between messages and the associated driving statistics or data will be later defined in 6.1.7.

4.4 Summary

Because the lack of vast production implementations of connected vehicles it is necessary to virtually simulate traffic flow through SUMO. The data that SUMO is capable of producing closely models that which the National Highway Traffic Safety Administration expect to become prevalent in the coming years [12, 56]. Through a SUMO simulation data will be

CHAPTER 5

Methodology and Experimentation

This chapter presents the methodology applied throughout this thesis. Section 5.1 examines the construction of the behavioral model, followed by Section 5.2, then behavioral pattern identification process. Section 5.3 presents the behavioral value and trust modification. Lastly Section 5.4 discusses the updating of the behavioral model. Figure 5.1 demonstrates the significant steps of the proposed behavioral model based trust management design implements.

5.1 Construction of a behavioral model

The steps that led to the creation of the behavioral model was divided into two distinct phases: data collection and the compile a behavioral model.

5.1.1 Data Collection

This phase will require the collection of data regarding the specific driving statistics of each vehicle within the localized geographical region. The focus is to collect data that is informative and will lead to a safer environment. More details about the data is provided in Section 6.1.6. The connected vehicle's will be the ones producing the necessary information, by using their own sensors they will be collecting their own driving statistics while simultaneously collecting information on their surroundings such as road conditions. This information will then be propagated to a nearby RSU for further processing. This is done to alleviate the connected vehicle from processing data

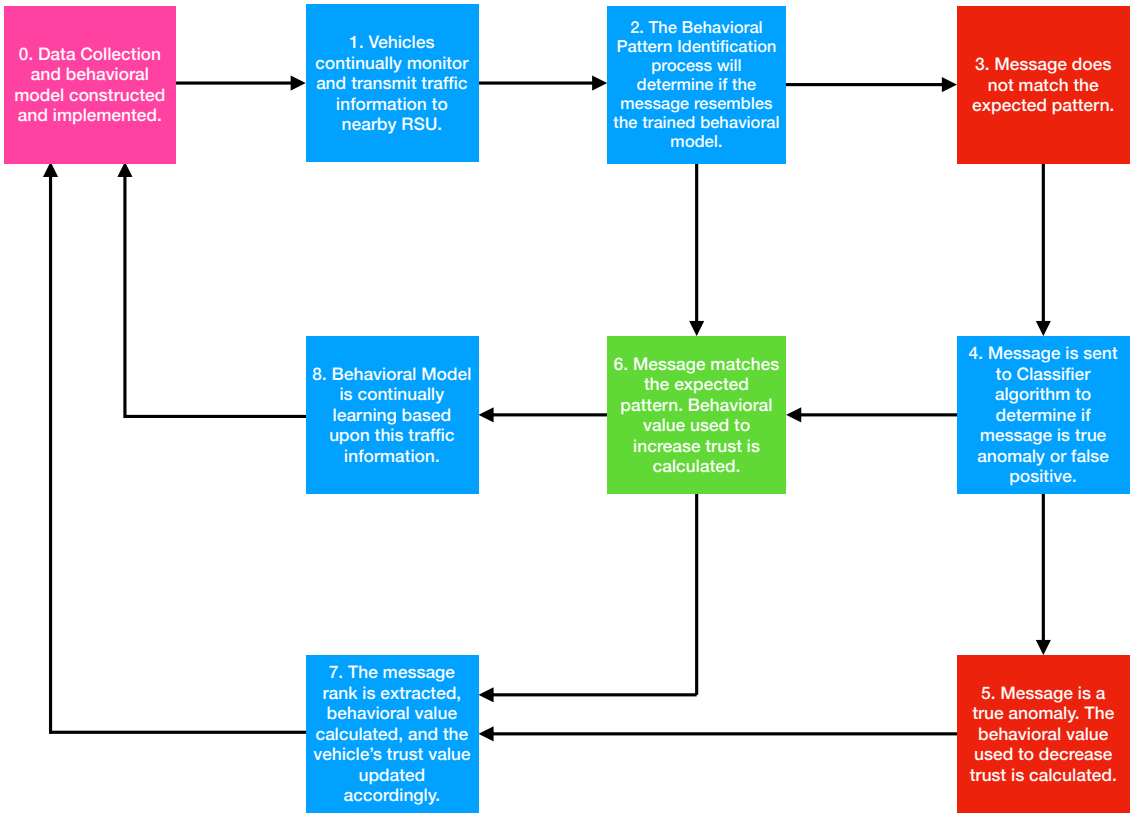

Figure 5.1 Graphical representation of each critical step describing how messages are evaluated and propagated through the Behavioral model based trust management approach. Pink represents the starting location, green represents a good behavior and red is the detection of a malfunctioning vehicle or malicious message

while allowing for the RSU to be able to conduct a deeper analysis of the collected data. Furthermore, this data collection process will be continually collecting new data, and this information can then be propagated through the system for evaluation at the later steps.

The driving statistics collected are listed below, with example data found in Figure 4.2

- **Vehicle position:** A value that corresponds to SUMO's representation of the road that the vehicle is driving on. While SUMO is limited to specific values associated with the map assigned to the simulation, real world data could potentially correspond to the address of the current road. Example: Main Street, Chattanooga, TN.
- **Speed:** In SUMO this corresponds to the instantaneous speed of the vehicle in meters per second, however, this could be adjusted based upon the units of measurements used.
- **Longitude and Latitude:** This corresponds to the GPS X and Y coordinates on the map that was used in the SUMO simulation. This is used in conjunction with the position such that a more accurate location on the road can be obtained.
- **Acceleration:** The instantaneous acceleration of the vehicle at the time of sending the message (m/s^2)
- **Motion State Speed:** A value to supplement speed as this corresponds to the vehicle's speed during the time unit measured instead of the instantaneous speed when the message was sent. This feature gives a more holistic view into the vehicle's speed without
- **Braking Rate:** This feature is the value associated with the deceleration of the vehicle at a particular instance during the simulation (m/s^2).

The collection of these features and the additional information provided by the message rank, described in Table 4.1, provide the necessary information to build a behavioral model.

5.1.2 Behavioral Model Design

Each message that vehicles send containing the driving statistics is processed and grouped based on message rank. These categories of driving statistics will serve as the foundation for the behavioral model. By designing a behavioral model for each message rank, it allows for development of a more precise model that is necessary for evaluating messages to identify driving statistics that do not reasonably match the behavioral model. This processing of messages to form the behavioral model will be conducted by the RSUs that enable for the behavioral model to be specific to the area that the RSU is located.

5.2 Behavioral Pattern Identification

After the construction of the behavioral model, the implementation of the behavioral pattern identification process can be discussed. This again will be conducted by the RSUs as they will 1. be collecting the driving statistics of vehicles 2. house the behavioral model. This means that as vehicle's are transmitting data the RSU will be collecting this data and using the behavioral pattern identification process to compare against the behavioral model.

It has been shown that implementing two tiers of machine learning for detection can drastically increase the efficiency and accuracy of the models [43], thus for the behavioral pattern identification process, there are two distinct levels: anomaly detection and classification. The anomaly detection aspect is designed to flag any message that is a clear malfunction or malicious injection of data. Further, a classification algorithm is implemented such that any message that is detected as an anomaly is additionally verified to determine if it is a true anomaly or a false positive. It is critical that entire behavioral pattern identification process still maintains the real-time detection and mitigation of threats, thus a performance evaluation will be necessary to determine the final algorithms implemented.

5.2.1 Anomaly Detection

To detect threats in the system it is critical to monitor the data that vehicles are transmitting. To accomplish this, vehicles are required to send their driving statistics as well as a rank that is associated with what the vehicle is currently experiencing. Malicious vehicles will often inject data into the network with the purpose of causing havoc; while malfunctioning vehicles will not realize they are submitting inaccurate information and thus the detection of these intended or unintended threats in real-time is critical such that these devices can be punished and effectively removed prior to being able to cause any harm. To monitor this data, a machine learning approach will be implemented using the behavioral model as a baseline; any data that does not reasonably match the known behavior of the road will be flagged as inaccurate, resulting in the vehicle obtaining a lower trust. There are a number of reason that a message would be flagged as an anomaly, examples include the following:

- A malfunctioning sensor would produce incorrect readings such that vehicle would send data that does not match its driving pattern.
- A malicious vehicle could be attempting to inject data to have subsequent vehicles rerouted so it would have the road to itself.
- The vehicle could also be driving faster than the road allows for which would be a negative behavior and thus punishable.

Because the highly dynamic nature of connected vehicles and the messages they are capable of sending, it was decided that there will be multiple instances of the machine learning models where each would be uniquely trained on data corresponding to a specific message rank that the vehicle sends. This is to say that for the list of possible message ranks found during the simulations each have their own instance of a machine learning model that will flag any message that does not resemble the behavioral model that it was specifically trained on. By having these multiple machine learning models, each is able to be fine tuned for the data that it is operating on. The behavioral

model trained on car collisions will easily be able to detect driving statistics that do not match the criteria that is expected during a car accident.

This portion of the behavioral model trust management design is represented by step 2 in Figure 5.1 where if the new data matches the expected behavior then the system progresses to step 6, or if the new data was not a match it will progress to step 3.

5.2.2 Classification

To further aid in the detection of true anomalies a classification algorithm will be used to backup the results of the anomaly detection portion of the behavioral pattern identification process. As was previously discussed in Table 4.1 and will be further discussed in Table 6.1, each rank of message will be associated with a specific driving pattern such that if the behavioral pattern identification process detects an anomaly it will then send those driving statistics to the classifier where it will be able to accurately predict which rank the driving statistic matches. Using this classifier as a backup will increase accuracy of the overall detection of threats, and will help safeguard against false positives that anomaly detection algorithms are prone to generating. The classifier has the capability to relate the driving statistics with the rank and if this predicted rank matches the vehicle's reported rank then the message is deemed accurate and trustworthy so the vehicle's behavioral value can be increased. If the classifier predicts another rank that better matches the vehicle's driving statistics, then it is decided to be a true anomaly and the vehicle is punished.

The classification methodology is represented by step 4 in Figure 5.1. If the message is determined to be a false positive (where the anomaly detection algorithm made a mistake) then the system progresses to step 6, or if the message is deemed to be a true anomaly, the message progresses toward step 5.

5.3 Behavioral Value

When a vehicle first enters the system and the RSU receives that vehicle's first message and the vehicle will be assigned a neutral trust value of 0.5, this has been referenced in the literature and is the standard for trust management approaches [6, 7]. As the vehicle traverses the network it will be sending communications to the RSUs at predetermined time units. The RSUs will then evaluate these messages using the behavioral pattern identification process, which will indicate whether or not the message subscribes to the standard or expected behavior for the given location or incident. Based on the results of the behavioral pattern identification process the RSU will calculate that vehicle's behavioral value. The mathematical formula that will be used to calculate the behavioral value (B_v) is based on three critical values: the number of anomalies (A_c), the total number of messages (M_c), and the rank of the message (R_m) that the vehicle has sent. The number of anomalies and total messages will be used as a trustworthiness ratio, while the message rank (corresponding to Table 4.1) will be used as a coefficient to dynamically change the rate that the trust is increased or decreased. Further, the message rank (R_m) used in the formulas below is dependent on the pattern identification and classification algorithms described above in Section 5.2.2. To sufficiently punish vehicles the rank used to calculate the behavioral value is the maximum value between the actual rank the vehicle sent or what the classification algorithm determined based on the driving statistics reported. By using the maximum value, malicious vehicles that under-report or over-report (send low ranking messages even if their driving statistics correspond to a higher ranking message and vice versa for over-reporting), will both be punished equally as severely.

The behavioral-based formula for increasing trust is in Eq. 5.1:

$$B_v = \frac{(M_c - A_c)}{M_c} \times \frac{R_m}{100} \tag{5.1}$$

This equation is used upon successful matching of behavioral patterns, seen in step 6 of Figure 5.1.

The behavioral-based formula for decreasing trust is in eq. 5.2

$$B_v = \frac{A_c}{M_c} \times \frac{R_m}{10} \tag{5.2}$$

This equation is used when the behavior is determined to be an anomaly, as shown in step 5 in Figure 5.1.

As discussed in Section 3.1.1, it is often harder to gain trust and easy to lose trust [39], implementation of this is presented in equations 5.1 and 5.2. By implementing the trust generation and modification with this strategy it offers real-time mitigation of threats because the idea that trust is easily lost.

The behavioral value must be able to appropriately remove vehicles that pose serious threats to the system. To further combat continuously malfunctioning or malicious vehicles, there is a mechanism that can administer a more severe punishment when necessary. To accomplish this, the RSU initializes an anomaly counter for each vehicle such that on every third inaccuracy the vehicle will be punished with a three times multiplier for that specific message. The associated behavioral-based formula is modeled below:

$$B_v = \frac{A_c}{M_c} \times 3\frac{R_m}{10} \tag{5.3}$$

5.3.1 Consensus Mechanism

As discussed in 3.1, trust management systems require a majority consensus as the evaluation of the data vehicles are sending. However, this is a problem as there are threats that target consensus such as a colluding attack. While this implementation can alleviate potential collusions through the detection of anomalies, instances where vehicles create the illusion of incidents (described in our threat model in section 2.4), will require a new consensus mechanism However, to combat traditional majority consensus approaches, an implementation of a δ (delta) time unit will

be used such that any message can be disproved through a majority consensus, however, this must occur within delta time units. Upon receiving the first record of an incident, the local RSU will initialize the delta, at that time any vehicles that approach the same position (the road location) that the incident was first reported has the potential to disprove the initial message. After the end of delta time, the RSU determines which report has the majority is decided to be accurate and so those that sent inaccurate data will have their trust values retroactively punished.

Retroactively punishing vehicles must be taken more seriously than the typical form of punishment. Injecting illusions has more potential to cause harm in the system because vehicles have the ability to bypass the anomaly detection aspect, thus have its message accepted as real and accurate data until otherwise proven. To accomplish this the punishment formula described in 5.3 is given a $3\times$ multiplier such that:

$$B_v = \frac{A_c}{M_c} \times 3\frac{R_m}{10} \tag{5.4}$$

5.3.2 Trust Modification

As previously mentioned the behavioral value will be the instantaneous representation of the vehicle's behavior, however, this behavioral value must be integrated with trust to achieve a more accurate representation of the vehicle's trustworthiness. After the determination of whether or not the vehicle has matched the expected behavior or is a true anomaly (such that it is either malfunctioning or malicious), the RSU will increase or decrease the vehicle's trust value accordingly, as represented in step 7 in Figure 5.1

The increase or decrease in the trust value of a vehicle is based on the behavioral value, as described in equation 5.5.

$$T_{new} = T_{cur} \pm B_v \tag{5.5}$$

Lastly, after a 24 hour period the RSU will reset each vehicle's trust value such that vehicles that were malfunctioning will not be forever punished, and conversely vehicles that were trustworthy for a substantial period of time will not be held in too high of a regard to dismiss if at any point it begins behaving maliciously.

5.4 Retraining process

Retraining of the behavioral model is necessary to prevent it from becoming stale or outdated. When each new message is received by the RSU, the messages from the vehicles are propagated through the system as described in Figure 5.1, such that the behavioral pattern identification process, behavioral value calculation, and trust modification can occur. After, the messages are compiled together again and appended to the appropriate training set such that the behavioral model is continuously learning the behavior of vehicles. Only messages that were determined to resemble the current pattern are appended to these training sets (as seen in step 8 of the same figure), as anomalies have the potential to contaminate the behavioral model and thus negatively influence the accuracy of the behavioral pattern identification process. Further, if patterns in the geographical area change, it should not be a drastic change but instead a slow progression that enables the behavioral patterns to be adjusted over time without the need for complete retraining of the behavioral models.

At the start of each new day, the RSU will undergo re-training of the behavioral model such that the new data captured from the previous day can be implemented into the current behavioral model to prevent the model from becoming old.

5.5 Illustrated Example

The purpose of this section will be to provide an illustrated example such that a cohesive description of the workflow can be demonstrated. Figure 5.2 shows the example simulation where vehicle 75 is the trustworthy vehicle and vehicle 30 is malicious sending false messages.

timestep_time	vehicle_id	vehicle_pos	vehicle_speed	vehicle_x	vehicle_y	motionState_acceleration	motionState_speed	brake_rate	Message
0	75	15.9	11.84	2296.67	2273.62	0	0	0	1
0	30	19.89	6.57	1673.62	1969.15	0	0	0	1
1	75	28.09	12.19	2285.51	2278.52	2358	236	0	3
1	30	27.96	8.07	1666.17	1972.24	2206	221	0	6
2	75	39.52	11.43	2275.04	2283.13	1581	394	0.76	3
2	30	37.85	9.89	1657.03	1976.02	2265	447	0	6
3	75	50.31	10.79	2265.16	2287.47	1557	550	0.64	3
3	30	49.67	11.82	1646.11	1980.54	2573	704	0	6
4	75	5.9	8.18	2253.79	2295.43	2059	889	0	3
4	30	62.8	13.13	1633.98	1985.57	2567	961	0	6
5	75	7.89	10.42	2257.52	2305.11	2572	1146	0	3
5	30	74.93	12.13	1622.77	1990.21	-2102	751	1	6
6	75	19.37	11.48	2262.11	2315.63	2420	1388	0	3
6	30	87.6	12.66	1611.07	1995.05	2018	953	0	6
7	75	31.42	12.06	2266.93	2326.68	1320	1520	0	2
7	30	99.52	11.92	1600.05	1999.62	2281	1181	0.74	6
8	75	3.73	11.94	2271.7	2337.62	-2227	1297	0.12	1
8	30	3.08	6.19	1591.84	1999.44	1494	1738	0	6
9	75	5.72	11.49	2276.3	2348.16	-4499	847	0.45	1
9	30	11.24	8.16	1599.39	1996.32	2094	1947	0	6

Figure 5.2 Example simulation data that demonstrates two vehicles, one trustworthy and the other malicious, interactions in the system and how their associated trust values will be modified based upon the messages sent

5.5.1 Trustworthy Vehicle

First we will start with the trustworthy vehicle, vehicle 75. At time 0 the vehicle enters the network and sends a message, when the message is sent the RSU will see that it has not seen vehicle 75 before and will first initialize this vehicle in the network and assign it a neutral trust value of 0.5. Then the RSU will begin processing the message that was sent. The RSU will extract the 'Message' field from the vehicle's message and will see that the rank is 1. This prompts the RSU to verify the driving statistics of the vehicle by sending the communication message through the anomaly detection algorithms that are trained on driving statistics with rank 1. Since this is

the first message this could be because the vehicle' being turned on and providing a trivial alert stating that it is raining. Since this is the trustworthy vehicle the behavioral pattern identification process verifies the message as accurate and thus the RSU begins the process of calculating the behavioral value using the equation for increasing behavior (Eq. 5.1). This is the first message that the vehicle sends and thus the current message count for that vehicle is 1, the message was evaluated as trustworthy thus the anomaly count is 0, the message rank for this specific message is 0. Thus using the increasing behavior equation the calculation is described as:

$$B_v = \frac{(1-0)}{1} \times \frac{1}{100} \tag{5.6}$$

This gives us a behavioral value of 0.01. After this calculation occurs the RSU will then increase the vehicle's trust according to this behavioral value as described in equation 5.5. This provides us with the the following calculation:

$$T_{new} = 0.5 + 0.01 \tag{5.7}$$

Thus the RSU will now recognize that the vehicle has a trust value of 0.51 or 51%.

This process is continued when vehicle 75 send the next message at time 1. This time the message rank is 3. The RSU extracts this value and will send the communication through the pattern identification process that is trained on driving statistics associated with rank 3 messages and because this is the trustworthy vehicle it is determined to be accurate. This brings vehicle 75's total message count to 2, anomaly count to 0, and the current rank of 3 allowing for the behavioral value calculation shown below.

$$B_v = \frac{(2-0)}{2} \times \frac{3}{100} \tag{5.8}$$

Providing a behavioral value of 0.03 that is then applied to increase trust such that vehicle 75's trust value now is 0.54. This process will continue and all the messages will be evaluated to true such that at the end of the simulation, vehicle 75's trust value will be 73%.

5.5.2 Malicious Vehicle

Similar to the trustworthy vehicle, vehicle 30 in this case will be sending primarily malicious messages such that its behavior and trust values will be decreasing. Vehicle 30 enters the network and sends its first message of rank 1, but to demonstrate the usefulness of the trustworthy ratio we will allow this message to be accepted as an anomaly. So similar to the first steps in the trustworthy vehicle the behavioral value is calculated and the trust value of vehicle 30 is now 0.51 or 51%. Vehicle 30 at time 1 will send a new message that is of rank 6, the RSU will receive this message and will propagate the message through the behavioral pattern identification process trained on messages with rank 6 and the pattern will be deemed an anomaly and thus malicious and now the RSU will begin the process of calculate how to punish the vehicle. Thus the total message count for this vehicle is 2, anomaly count is 1, and the current message rank is 6. According to the behavioral value equation used to decrease behavior presented in equation 5.2, the behavior of the vehicle is calculated as follows.

$$B_v = \frac{(1)}{2} \times \frac{6}{10} \tag{5.9}$$

This provides the behavioral value of 0.3. then is then applied to decrease trust by the equation below.

$$T_{new} = 0.51 - 0.3 \tag{5.10}$$

This means at after one malicious message, the RSU will store vehicle 30's trust value as 21%. This process will continue and the next message the vehicle sends will also be determined to be an anomaly and the behavioral equation will now be:

$$B_v = \frac{(2)}{3} \times \frac{6}{10} \tag{5.11}$$

This provides the behavioral value of 0.4, used to decrease trust by the equation below.

$$T_{new} = 0.21 - 0.4 \tag{5.12}$$

This new trust value would be a negative number, however, the system will only decrease trust such that it is between 0% and 100% thus after two malicious messages vehicle 30's trust value is now 0%.

CHAPTER 6

System Evaluation

This chapter presents the necessary steps used for experimentation as well as the results from simulations that address the threat models described in Section 2.4. An analysis of the results follows each specific use case.

6.1 Experimentation setup – Simulation Setup and Design

The experimentation setup consists of loading a map layout, generation of a network file, spawning vehicles and mobility, and configuration of the SUMO file.

6.1.1 Loading a map layout

As discussed in Section 4.2, SUMO is capable of implementing a simulation on a OpenStreetMap (OSM) that allows for custom map to be selected. A map of downtown Chattanooga was selected for the simulation as shown in Figure 6.1. OSM is able to export manually a selected portion of the map, and will output this map to an `XML` file that contains information such as the ID of the road, the speed limits, and the latitude and longitude of the road.

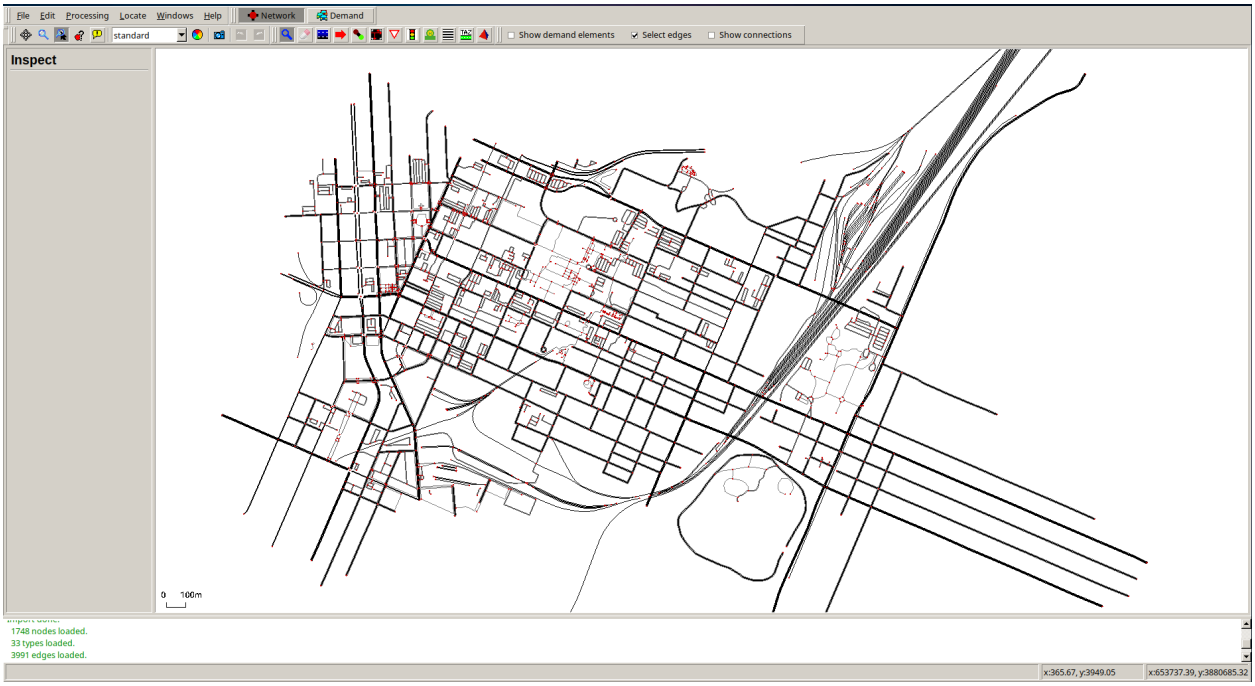

Figure 6.1 The map that SUMO used as the basis for simulating traffic mobility

6.1.2 Generating network file

Through the use of the SUMO suite of tools, it is possible to convert the OSM `XML` file into a network file that SUMO is able to use. This is done through the *netconvert* tool that reorganizes the OSM data into lanes with the associated length of the road, speed limits, latitude and longitude, as well as the edges of the lanes such that the intersection of lanes is preserved from the OSM data.

6.1.3 Generating vehicles and mobility

As described in Section 4.2, SUMO represents each vehicle in the network through a unique routing protocol that defines every step of the vehicles life in the simulation. This information is stored in two critical files, the trips and routes file. The trips file defines the vehicle's depart time (the time it enters the simulation), and the lane that it will enter and exit from while the routes correspond to each vehicles route that it will traverse through the simulation from the starting and

ending point as was defined in the trips file. Necessary to the creation of these two files is another tool supplied by the SUMO suite, `RandomTrips.py`, a python script that accepts the network file that was converted from the OSM file and an argument to represent the length of the simulation in seconds. The `RandomTrips` python script is able to randomly generate a number of vehicles, the starting and ending location, the departure time, as well as the route each vehicle will take. While the number of vehicles can be tuned, generally the random trips script will stagger the spawn of vehicles such that a new vehicle enters the system at each new time interval. While it is possible to manually generate the necessary files, this random trips script is able to automatically generate unique scripts without manually defining each and every vehicle's route.

6.1.4 Configuring the SUMO file

The last step necessary prior to the running of the simulation is the creation of the SUMO configuration file. This is how SUMO will recognize files such as the network, route, and trips files. In the configuration there is also the option to modify the length of time that the simulation will run, this time needs to correspond to the time that was used in the trips and routes generation so that all vehicles will successfully be able to exit the network upon simulation completion. If the random trips script was not used it is possible to define which routing algorithm SUMO would implement, the available algorithms include numerous graph traversal algorithms [59]. However, with the ease of the random trips script, this configuration variable was not set.

SUMO allows for two forms of simulation: a back-ended process that is done via the command line, as well as a GUI that can be used to visualize the simulation and see each vehicle and the routes taken. An example of a GUI simulation is shown in Figure 4.1. The GUI has the ability to slow down the simulation as well as step through each time step of the simulation. SUMO begins by processing the routes and trips file such that the vehicles will enter the system at the specified road and time. SUMO then linearly progresses by incrementing the time, and thus

spawning a new vehicle, as well as progressing each vehicle in the system at the OSM and SUMO defined appropriate speed along the route defined in the routes file.

6.1.5 Simulation of Car Collisions

SUMO does not currently have the capabilities of implementing collisions, thus a method to imitate a collision in SUMO was essential to the success of this work. It was decided that through placing stop lights throughout the map that was used in the simulations would be a sufficient representation of how vehicles approach a car collision. Vehicles will approach a stop light and if the light were red, the vehicles would recognize this and would hit the brakes to come to a stop. After a specified amount of time the stop light would change green and the vehicles would continue on their way.

A smaller and more trivial simulation was designed and implemented to collect initial data that enables an analysis such that the main simulation would successfully implement stop light to imitate a car collision. This initial simulation was conducted on a simple intersection with only 1 to 10 vehicles where the light would be red for 5 time units. This allowed for vehicles to approach the light, apply the brakes to come to a complete stop, then continue on their way.

6.1.6 Data Collection and Processing

The data collection process is vital to the evaluation of this approach as without the necessary data to accurately describe the vehicles behavior the machine learning models used will be limited in their ability to develop a precise representation of the behavioral model.

As stated in Section 4.2, SUMO is capable of producing several different output files each corresponding to a specific type of data such as vehicle emissions or raw position. Specific to this thesis the following SUMO command was executed to obtain outputs for a single simulation:

```
sumo -c map.sumocfg --fcd-output FCD-Trace.xml  --lanechange-output\
 LaneChange.xml --device.ssm.probability 1 --amitran-output Trajectories.xml
```

The Floating Car Data (FCD) output contains the location (latitude and longitude) and speed for every vehicle in the network at every time step, this output is similar to that of how a GPS would report data [56]. The lane change output contained information specific to the lane that the vehicle was currently on, the ID of the road at each timestamp [56]. The `amitran` output is in regard to the trajectory of the current vehicle. This include the `motionState` speed of the vehicle as described in Section 4.3. The trajectory also included the acceleration of the vehicles at each timestep. Lastly is the inclusion of a surrogate safety measures (SSM) device, discussed in Section 4.2. The SSM device is attached to 100% of the vehicles and is implemented to report the braking rate of each vehicle at every instance. The SSM device that each vehicle has produces its own output that is unique to every vehicle.

SUMO outputs these files into an `XML` format, that are then converted into a `CSV` file using a companion python script (`XML2CSV.py`) that comes with SUMO. To compile these results into a single file, we used the `Pandas` library [60], that facilitates processing and joining these different output files into a single standardized and uniform output that represents the entire simulation.

6.1.7 Creating the distinction between messages

To simulate these messages there needed to be a distinction between the rank and what the vehicle driving data would show. Table 6.1 shows this distinction. Some of these values are specific to the SUMO simulation and there may be overlap between these messages. This is attributed to the lack of: 1. real world information such as rain detection data 2. SUMO's capabilities are limited such that it is not possible to implement sensor data that connected vehicles require.

Table 6.1 Relation of rank to the expected driving pattern

Rank	Driving Pattern
0	Vehicle must be in motion: The acceleration must be less than 1450, and the vehicle should not be experiencing braking.
1	No major braking (0-1): The acceleration must be less than 1450.
2	No major braking (0-1): Vehicle must be in motion. Acceleration must be less than 1450.
3	No major braking (0-1): Vehicle must be in motion. Acceleration must be greater than 1450.
4	Braking occurs (1-2): Vehicle speed must be less than 1200.
5	Heavy braking (2-3): Vehicle speed must be less than 800.
6	Major Braking (3-4.5): Vehicle speed must be less than 400.

Upon running over 500 simulations the overall average values of the the features were obtained as well as the average maximum value. The maximum values were divided in half and averaged again with the actual average values. This combination of half of the maximum and average values provided a psuedomedian value that is used as the foundation for division between messages and the associated ranks. This lead to the psuedomedian of acceleration to be calculated to 1450, speed to be 1200, and braking rate to be less than 1. This psuedomedian value is necessary to represent the messages and ranks that vehicles would be sending and be able to create a clear distinction that will be used later for our comparison to detect misbehaving vehicles.

An example would be any rank/message that corresponds to vehicle's not having much of an impact while driving must have an acceleration of less than 1450. Rank and message 3 on the other hand corresponds to a change in driving pattern such as speeding up or changing lanes, to create a distinction with this message between the others, this message must have an acceleration higher than the average of 1450.

Using the car crash simulation, incident were able to be simulated such that a collection of driving statistics were collected. These driving statistics were then divided into three separate categories as was described in Table 6.1. This division was necessary such that there would be a clear distinction in the type of incident and the model that would represent said incident. By clearly defining the differences in critical messages it will better aid in the latter behavioral pattern identification process.

After each SUMO simulation was conducted each line of the standardized output was processed, and if the data matched the message distinction defined above in Table 6.1, the corresponding message was appended to the message. With this, the simulation has now successfully simulated live V2V communications complete with sensor data such as GPS, safety devices, and a message to alert of vehicles or infrastructure of the current road or environmental conditions.

6.1.8 Injection of Anomalies

SUMO simulations are a method that enable study of traffic mobility, because of this the ability to simulate a vehicle that behaves abnormally is outside the scope of the SUMO implementation. Thus in order for the simulation to have malicious vehicle, the data collection process is slightly altered such that anomalies are injected into the data. This is primarily done through the modification of the message rank. Because the distinction of messages from Table 6.1, any message rank that does not meet those characteristics should be considered an anomaly and the vehicle either malfunctioning or behaving maliciously. The method for injecting anomalies is unique such that each use cases that addresses the specific threats described from Section 2.4.

6.2 Machine Learning Model Performance Evaluation

Evaluation of the machine learning models is necessary such that the implementation designed meets the real-time needs for detection and mitigation in threats that connected vehicles require. To accomplish this evaluation different machine learning algorithms were implemented for both the anomaly detection and the classification portion of the behavioral pattern identification. For these evaluations the same training and testing data was used to obtain results that can be compared for accuracy and time to process. The distribution of messages in the training and testing data for the machine learning algorithms is shown in Tables 6.2 and 6.3 respectively.

6.2.1 Anomaly detection

For implementation of the anomaly detection model, three algorithms were tested to determine which offered the best accuracy and performance such that threats could be detected while also maintaining the real-time demands of such a system. Robust Covariance Elliptic Envelope, Local Outlier Factor (LOF), and Isolation Forest were chosen as these were models previously shown in the literature [42, 61]. But further, Coleman et al. showed how these different models compare in terms of accuracy in anomaly detection showing that the Isolation Forest was best able to detect threats for a very smaller feature set. While Garcia et al. showed that using differently sized features sets provided different accuracy's for each model and thus re-testing of these models will be necessary to determine which model will provide the highest accuracy for the selected feature set. Using the default parameters, each of the models were trained on the same data and tested on the same simulation. Accuracy and time to process results can be found in Table 6.4.

With these results described in Table 6.4, it is clear that elliptic envelope machine learning algorithm was the most accurate but also able to process messages in 67% of the time of LOF, and 8.9% of the time it took Isolation Forest to process the message. Since elliptic envelope has the highest accuracy and the fastest processing time, it is the best choice for this implementation as it meets both the accuracy and real-time requirements that connected vehicles necessitate.

Table 6.2 Distribution of message ranks in the machine learning training data set. These messages were obtained from over 500 simulations with each simulation containing 300-700 vehicle. The simulation was conducted for a period of 86,400 seconds (one day)

Message Rank	**Number of Data Items**
0	84,956
1	44,329
2	43,491
3	12,735
4	41,599
5	11,250
6	45,611

Table 6.3 Distribution of message ranks in the machine learning testing data set. This data was obtained from a single simulation that contained 100 vehicles, and also simulated 1 day of driving for each of the vehicles

Message Rank	Number of Data Items
0	1,035
1	2,456
2	2,015
3	1,790
4	52
5	17
6	75

Table 6.4 Performance evaluation of each of the anomaly detection algorithms

Machine learning Model	Accuracy	Time to Process Message (seconds)
Elliptic Envelope	89.3%	0.00258
Local Outlier Factor	80.9%	0.00385
Isolation Forest	73.6%	0.02912

6.2.2 Classification

For this implementation three models were tested to determine which offered the best accuracy to processing speed such that threats could be detected while also maintaining the real-time demands of such a system. Decision trees, K nearest neighbors (KNN), Linear Support Vector Machine (LSVM), Multilayer perceptron (MLP), Random Forests were chosen as these were models as it has been shown in the literature to have provide higher accuracy of results, reduced false positive ratings, and require the least amount of time to process the data in an IoT environment that is absolutely critical for the needs of this thesis [48, 49]. Further, these algorithms are also readily accessible through python libraries that enable testing such that the machine learning models could be dynamically applied [2]. Using the default parameters, each of the models were trained on the same data tested on the same simulation. Training time, accuracy, and time to process results can be found below in Table 6.5.

The time to process these messages will be used in conjunction with the Elliptic Envelope

anomaly detection, so to account for this the time to process each message was calculated through the subtraction of the time to process the message with only anomaly detection, 0.00258s according to Table 6.4.

According to Table 6.5, decision tree and random forest offer the significantly better accuracy (96% and 97% respectively) than the other models tested, while LSVM offered the fastest time to process the message at 0.00170 seconds, however, in a close second decision tree's time to process the message was 0.00172 seconds. Decision trees offer a high accuracy that will aid in the detection process, while also supporting the real-time demands of smart cities and connected vehicles by having the second fastest time to process messages of 0.00172 seconds.

Table 6.5 Performance evaluation of each of the classification algorithms

Machine learning Model	**Time to Train**	**Accuracy**	**Time to Process Message (seconds)**	
Decision Tree	1.39871	96%	0.00172	
KNN	0.31830	74%	0.00236	
LSVM	197.15077	52%	0.00170	
MLP	153.94936	79%	0.00180	
Random Forest	33.49497	97%	0.00835	

6.2.3 Summary

The true implementation of the behavioral pattern identification system uses both the anomaly detection algorithm as well as the classification algorithm together to enable better detection and mitigation of threats. Upon analysis of each individual machine learning model it was found that elliptic envelope offered the best accuracy (89.3%) and time to process each message (0.00258 second) for the anomaly detection portion. While decision trees offered the best performance for the classification algorithms with an accuracy of 96% and a time to process an individual message of 0.00172 seconds. Thus to evaluate the performance both algorithms were used together to determine the accuracy of the overall model. The same training and testing data was used as previous evaluation methods and the accuracy metrics are described in Table 6.6

Table 6.6 Performance evaluation of the behavioral pattern identification system using both Elliptic Envelope and Decision Trees machine learning models

Message Rank	**Number of Anomalies**	**Accuracy**
0	89	91.4%
1	114	95.4%
2	172	91.5%
3	0	100%
4	0	100%
5	0	100%
6	0	100%

The results show that the implementation of the behavioral pattern identification system offers increased accuracy that the anomaly detection algorithm alone was not able to achieve. Further, it is shown that identification of all incident messages had a 100% accuracy rating showing that the ability of the behavioral model to accurately represent these messages further allows the pattern identification process to detect and mitigate these threats. In a simulation with 289 vehicles, a total number of messages sent from all the vehicles was 7,440, this provided the results described above that led to an average of 0.00430 seconds for processing a single message[1]. This was based on the average results of running a day's simulation with 289 vehicle and a total message count of 7,440 across the entire day, which will allow for 233 messages per second. This level of performance means that if a vehicle sends one message every minute, this implementation would be able to sustainable process of nearly 14,000 vehicles' data and driving statistics

6.3 System evaluation under the threat model

To evaluate the system, use cases were designed in such a way to demonstrate the capabilities of this approach as well as imitate the threat models that were described above in section 2.4. The use cases are divided into two sections, the first being the detection and mitigation of malfunctions or obvious anomalies such as a vehicle driving on the interstate at 70mph but report-

[1]This performance resulted from a late model AMD Ryzen 1700X x86-64 processor at 4.2Ghz.

ing a wreck without any drastic changes in driving patterns. Each of these simulations contains 5 vehicles where each vehicle remained in the system for 50 time units (minutes) transmitting their driving statistics and messages at the top of the minute. The second area of uses cases involves the malicious actor creating the illusion that the reported event actually occurred. This is to say that the vehicle's driving statistics show that the event occurred and it was not detected as an anomaly, but upon consensus from the near by vehicles it is apparent that the event did not occur. These simulations also were ran with 5 vehicles where each of the vehicles remained in the system for either 20 or 25 minutes, also transmitting their driving statistics and messages at the top of the minute.

6.3.1 Use case: Good vehicle

This use case was designed to simulate all vehicles operating in a trustworthy fashion. All vehicles will enter the simulation at the same time and will transmit the message associated with their driving pattern (as described in Table 6.1) without any modification. This is the foundational simulation that demonstrates how the standard vehicle will gain trust in the system, when there are no other malicious vehicles present.

6.3.1.1 Analysis

We can see that all vehicles trust steadily rise in the system. Because the vehicle's are primarily sending warning messages, that have lower ranks their trust increases slow but steadily. Vehicle 5 is the first to reach a 100% trust value after 28 messages, while at time 31, vehicle 3 had only reached 95% trust. We can see that vehicle 3 had some messages that were detected to be an anomaly (such as time 17, 27, 30) but because of the lower ranking messages it was not a significant impact to the overall trust value.

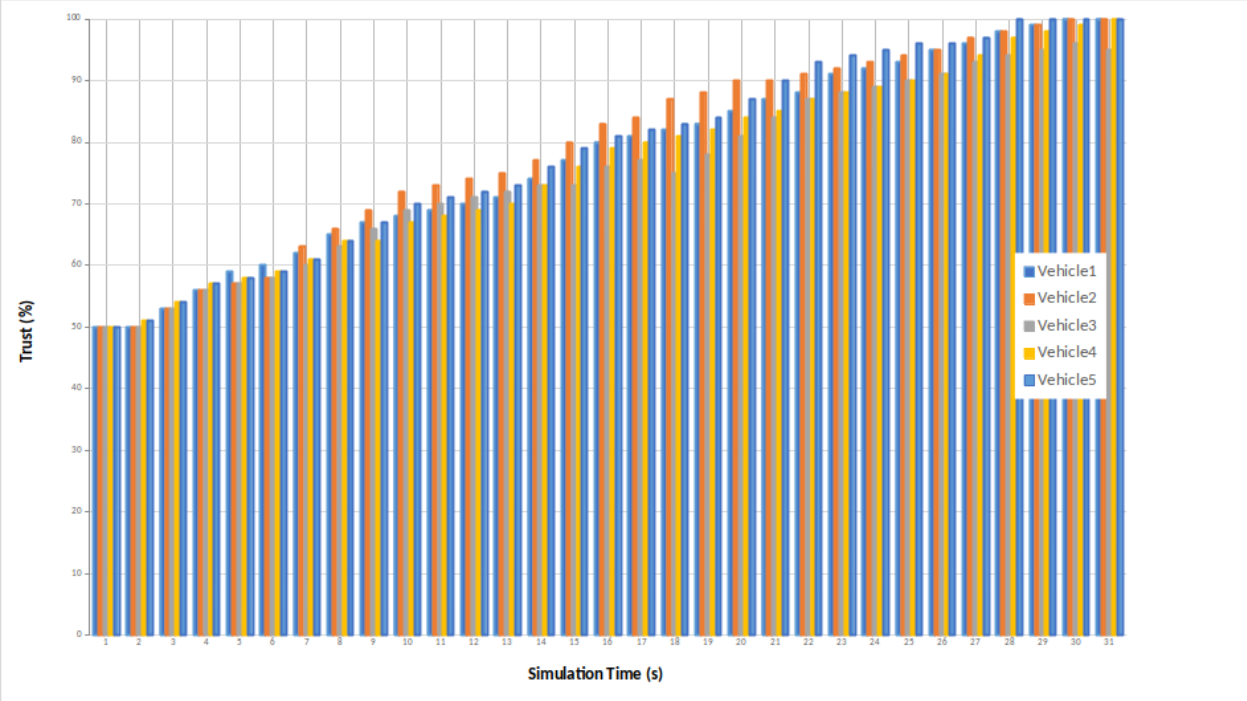

Figure 6.2 Trust value results from simulation with 5 vehicles accurately reporting driving statistics

6.3.2 Use case: Bad Vehicles

This bad vehicle use case illustrates how trust decreases in the system when the vehicles behave maliciously. All five vehicles in this simulation enter the system at time 0 with a trust value of 0.5 and begin driving through the system exactly as use case 1 (implying a standard driving pattern), however, instead of the correct messages that were described in Table 6.1, each vehicle will send malicious messages were each vehicle will correspond to a specific message. This is to say that vehicle 1 will be sending false messages that correspond to a 1, while vehicle 2 will be sending false messages with a rank of 2, etc.

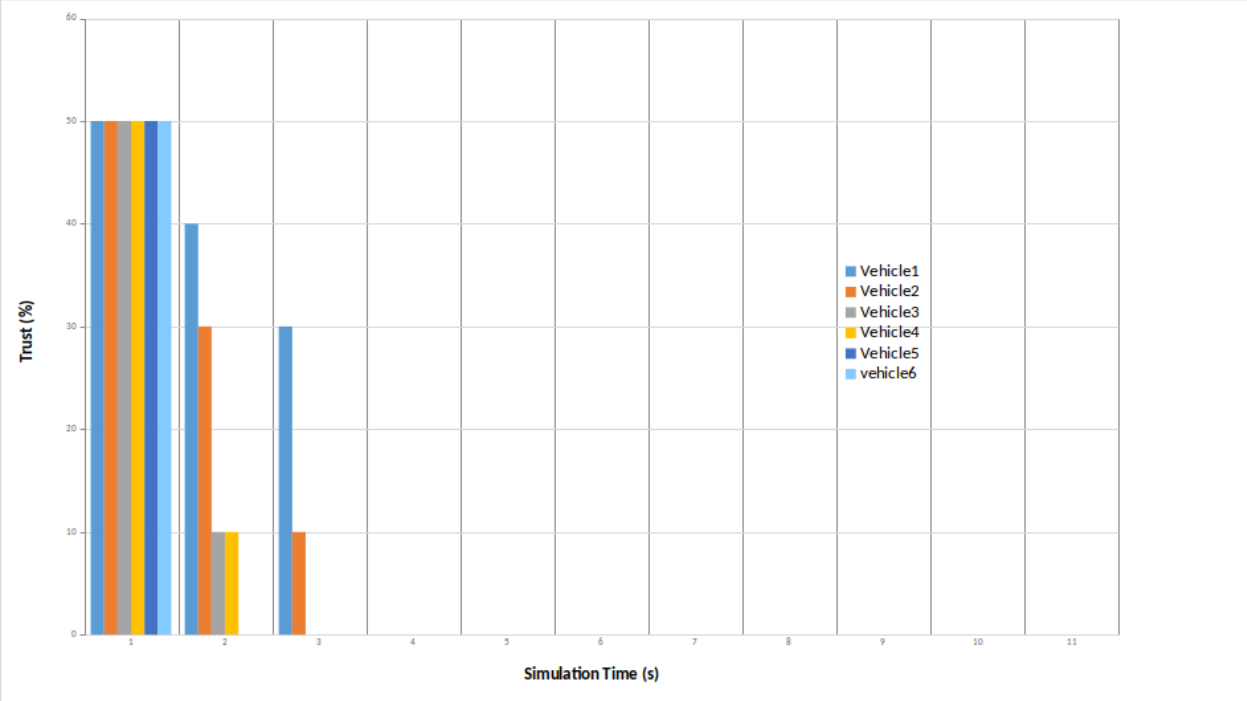

Figure 6.3 Trust value results from simulation with 5 vehicles maliciously reporting driving statistics

6.3.2.1 Analysis

It can be seen that the first message that the system receives is determined to be an anomaly and the vehicles are all punished. Because of the high-ranking messages that the vehicles send, their trust values are negatively effected as was discussed in 5.2. This shows that the system is able to effectively mitigate new malicious actors instantaneously such that they are not able to negatively affect the system. These results show that the detection and mitigation properties of this system are able to operate in real-time.

6.3.3 Use case: Breakout Fraud Threat Simulation

This use case will be used to show how the system reacts during attempted breakout fraud as described in Section 2.4. Vehicles will send accurate and trustworthy messages up until a certain point at which the same vehicles will begin acting malicious sending false messages through the remainder of the simulation. This use case is important because it will demonstrate how quickly the system is able to punish those vehicles that have been accurate leading up unto that point. This is necessary because vehicles that have high trust will have more influence and thus be relied upon more and if they are to begin acting maliciously it could cause serious harm.

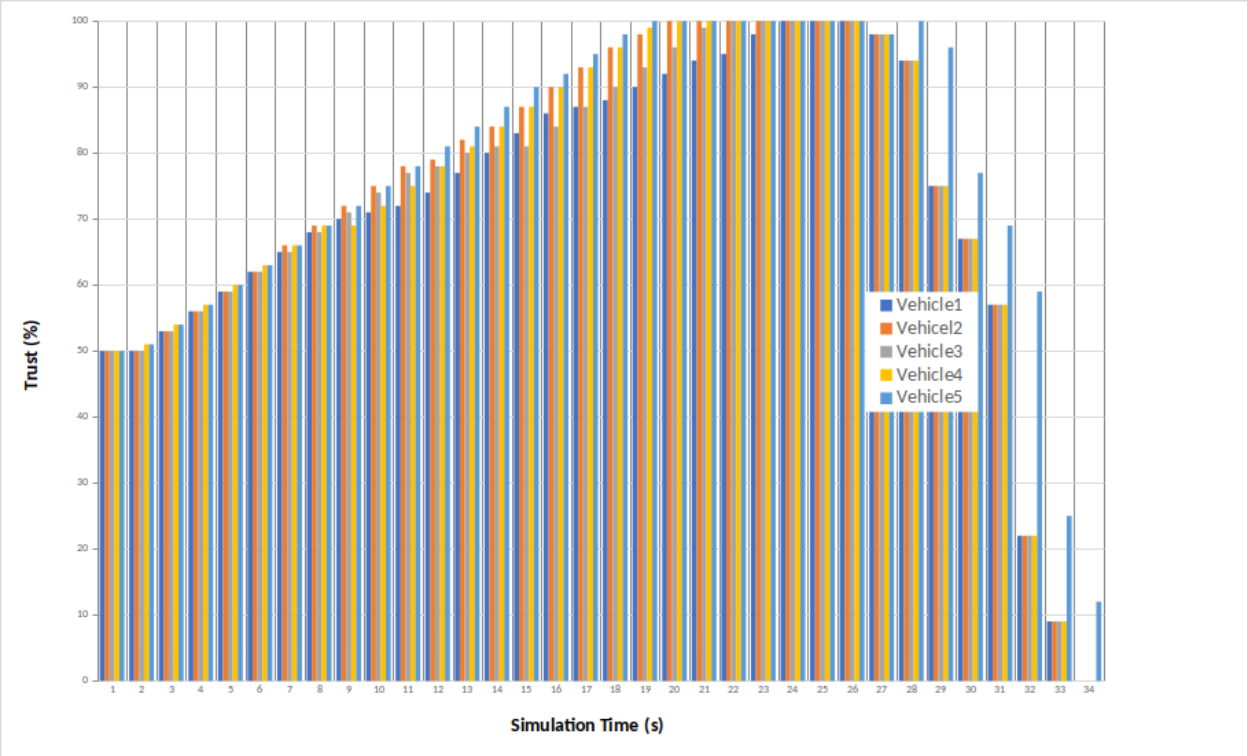

Figure 6.4 Trust value results from simulation of vehicles committing breakout fraud as described in section 2.4.1. Where vehicle send messages to gain trust, then begin sending malicious messages

6.3.3.1 Analysis

The results clearly show that each of the five vehicles are sending accurate and trustworthy messages the first half of the simulation as the results mimic that of the use case 1: the good simulation. But it can be seen that at time 26, the vehicle's trust begins dropping, by timestamp 30, only 4 messages later the vehicles went from a 100% trust to 40% effectively removing themselves from the system. Just another 3 messages later at timestamp 33 the vehicles are at below 10% trust values. In just 8 minutes the vehicles went from having achieved the highest possible trust to having earned the lowest trust of 0%. It can be seen that through the trust formula equations 5.1 and 5.2, there exists some headroom for vehicles that have a high trust factor. But this is to be expected as those vehicles have proven that they were trustworthy should not be hastily punished for sending messages that could be because of a simple misreading of the sensor.

6.3.4 Use case: Comeback Simulation

A comeback use case is to simulate when vehicles experience malfunctions in sensors that would cause them to send inaccurate data thus impacting their trust negatively. But at any point the sensors becomes fixed and the vehicle begins sending accurate data. To simulate this, vehicles will send inaccurate messages the first half of the simulation, then the second half the vehicles will begin sending accurate messages. This use case again shows how quickly trust can be decreased such that vehicles sending inaccurate data will be a threat to the system similar to that of use case with bad behavior, but it will also show how trust is regained after the problem is resolved. While aimed at malfunctioning vehicles, this use case also demonstrates how a malicious vehicle would regain trust in the system and how difficult it is to regain trust after it has been lost.

6.3.4.1 Analysis of the primary comeback simulation

This simulation begins with the all vehicles sending malicious messages for the first 25 messages. We expect this to mimic the results from use case 2, however, it can see that vehicle 5 experiences a gain in trust at time 1. This is an acceptable gain for these simulations because it's most likely because the vehicle reporting that it is stationary and holding the brakes that matches the criteria of a car crash. But upon all five vehicles reaching 0% trust value, they continue sending inaccurate messages that do not match their driving statistics thus decreasing their trustworthiness ratio even further. At timestamp 25 the vehicles begin sending accurate messages that does not impact their trust until a few messages later which is attributed to the low trustworthiness ratio. These results (Figure 6.5) also demonstrate that because vehicle 5's initial messages were accepted for the first few instances, it was able to increase its trust a full four messages sooner than the next vehicles were able to. But it can be seen that when the vehicles were gaining trust and sending accurate messages it took vehicle 5 to 10 messages before it was able to obtain 10% trust demonstrating that if a vehicle has been persistently untrustworthy it takes numerous messages to be evaluated in order to have a small effect on the trust values. Vehicle 1 also demonstrates that even if you are beginning to gain trust again then send an inaccurate message that the trust which took over 10 messages to gain 10%, can be lost in just one minor inaccuracy. As was discussed in 3.1.1 it is hard to gain trust but easy to lose trust, and whenever you lose trust it's even harder to gain it back.

In addition to the initial comeback simulation, there is also an additional version of this that is used to simulate malicious vehicles that realize that all other vehicles are disregarding its messages, it will begin to send accurate data in hopes to gain enough trust such that other vehicles would be impacted by its messages. But to gain trust quickly the malicious vehicle will only reporting incident messages that have a higher rank. The reporting of critical incident messages is done so as an attempt to gain trust as quickly as possible and while not sending other messages. This is important because again it will show how trust is regained in the system but through sparse

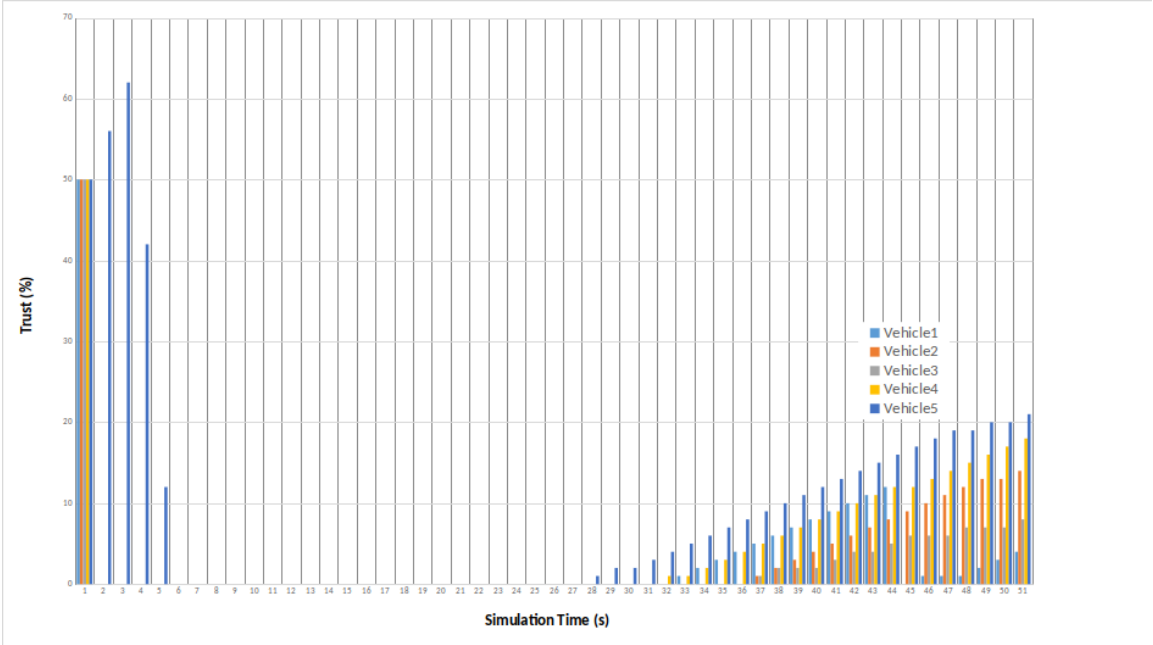

Figure 6.5 Trust value results from simulation of vehicles attempt to re-gain trust after losing it from sending several malicious messages

incident message evaluation instead of the warning messages that are not high ranking.

6.3.4.2 Analysis of the secondary comeback simulation

Here we can see that this simulation and results correspond closely to the initial comeback results. The first few messages passing as success is not great but can be attributed to the lack of additional sensor data that could've provided more insight into whether it was a wreck or just the vehicle being stationary after ignition. After that the results quickly show that the vehicle's trust value drastically diminishes down to 0% again. Then we can see that about half way through the simulation vehicles begin sending messages that were being evaluated to accurate and trustworthy. What we see now is that the vehicles go through comparatively steeper increases when compared to the previous comeback use case. However, with the sparsity of the incident messages these

steeper increases in trust only occur once every four of five messages. By using this tactic as an attempt to increase the trust of the vehicles is shown to be ineffective as each of these high ranking messages only increase the trust by at most 5 to 6% at a time. These results are presented in Figure 6.6.

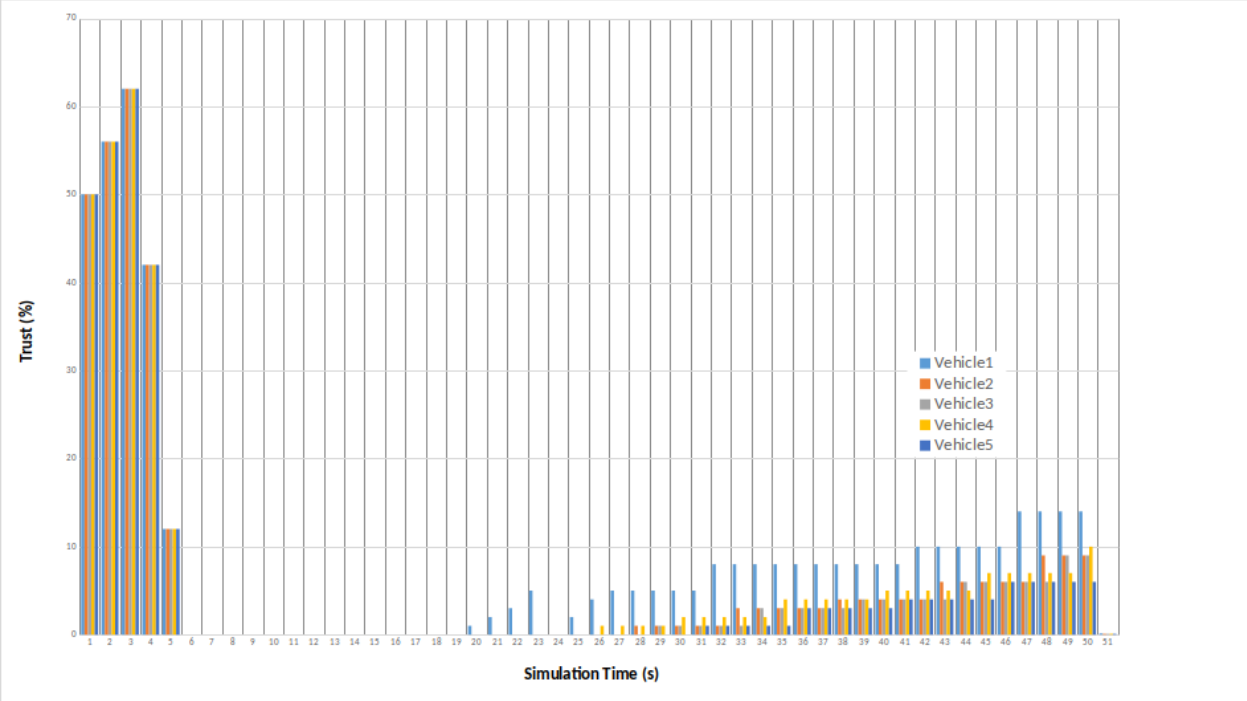

Figure 6.6 Trust value results from a simulation of vehicles attempting to quickly re-gain trust after losing it from sending several malicious messages

6.3.5 Use case: Selective Behavior (Flip-Flopping)

The Flip-Flop use case will be used to describe the trust of vehicles that will switch from sending good message to sending bad messages on a set schedule as an attempt to disrupt the system while simultaneously attempting to maintain a high enough trust to be listened to. This simulation also demonstrates how a vehicle's trust would be impacted if a specific sensor on the

vehicle were to begin malfunctioning. Since every message would not require every single sensor's data, one specific sensor could be malfunctioning that would go undiscovered for a few instances upon which it is required to send data. This data would cause the message to be flagged as an anomaly and thus the vehicle would be punished. To simulate this all vehicles will begin the simulation and then all simultaneously start sending good messages then flip to sending bad messages. This flip-flop of behavior could occur after any set amount of time, specifically three messages, but in a very predictable manner. In this case, the flip flop decision of three messages was selected as it would provide enough occurrences where it is easily displays that as time progresses trust dynamically becomes easier to lose while harder to gain.

6.3.5.1 Analysis

As stated above, this simulation involves the first three messages being sent to be accurate and trustworthy, in the results it can be seen that the trust increases during these three messages before the vehicle begins sending false data. Because of the nature of the data being sent the trust value is drastically decreased to 0% after just another three messages. The increase in trust values between the vehicles ranges between 2% and 6%. But those vehicles that do not send higher ranking messages such as vehicle 3 and vehicle 4 require three messages to obtain a 2 or 3% trust value and just a single message to drop the trust value back to zero. Because of the drastic decrease capabilities that the implement behavioral-based formula if these results are detected they can most likely be attributed to a malfunctioning sensor. These results are demonstrated in Figure 6.7.

6.3.6 Use case: Selective Behavior (Mixed)

Mixed behavior is an extension of Flip-Flop but instead of being in a set schedule, a vehicle will spontaneously send a false message at any point in the simulation. The flip flop use case

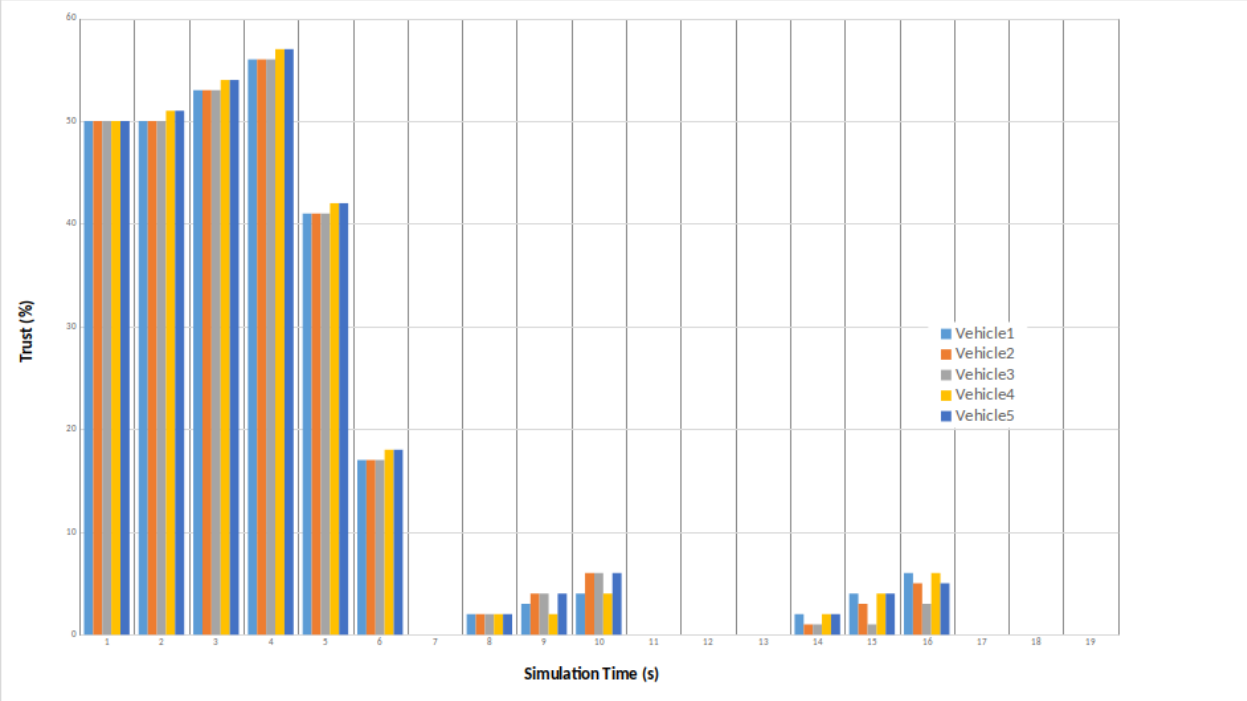

Figure 6.7 Trust value results from a simulation of vehicles the frequently switch from sending accurate messages to malicious messages

can be used to derive another involving mixing of behaviors. This form of mixed behavior can be attributed to any number of scenarios such as a malicious vehicle attempting to build trust to where it can send a malicious incident message and then return to sending good messages such that it retains a high trust value. This use case can be attributed to that similar to the flip flop a specific sensors on a vehicle being broken that whenever it has to report data causes the trust implementation to punish said vehicle because the data is inaccurate. This use case will be simulated by a vehicle sending false messages but at first sending inaccurate message sparsely but as time progresses the vehicle will be sending primarily false messages.

Another use case will be the completely mixed behavior. The vehicles will randomly decide when to send malicious messages or the vehicle would experience a degradation of a critical sensor such as a speedometer. This use case will demonstrate how the system is managed when the

malicious devices begin acting in a random fashion.

Both of these use cases are critical as they show how trust is impacted by a malfunctioning sensors or malicious vehicles sending inaccurate messages as the time in the simulation proceeds.

6.3.6.1 Analysis of the primary mixed behavior simulation

Here it can be seen that the vehicles begin the simulation with an increasing trust value. Then at time 5, all vehicles will send an inaccurate message that are evaluated to be an anomaly. This one anomaly takes their trust from 62% to 53% (because of the trustworthiness ratio we do not expect to see their trust value be completely diminished). After this one anomaly, the vehicles will continue again with sending five accurate messages. Increasing their trust 15%, followed by now two anomalous messages thus decreasing their trust to a range of 9-16%. These two additional anomalies have changed their associated trustworthiness ratio from 1:10 to 3:13 that demonstrate a decrease in trust by approximately 50% effectively removing them from consideration as their trust does not meet the criteria to contribute to the decision making process. With the now higher trustworthiness ratio the trust values of the vehicles are not capable of rising above 10% in the five messages sent. It can be see that after the trustworthiness ratio increases to nearly a quarter, anomalies are drastically impacted such that only one anomaly is able to diminish any trust the vehicles were able to build during that time. These results are demonstrated in Figure 6.8.

6.3.6.2 Analysis of secondary mixed behavior simulation

Because of the randomness of this simulation results displayed have a significant difference in other use cases shown in this work. Each of these vehicles can be explained through a number of reasons. Vehicle 1 has initial decreases in trust and by time 3 is already down to almost 30%. But after another thirty messages with only a few minor complications vehicle 1 is able to achieve a

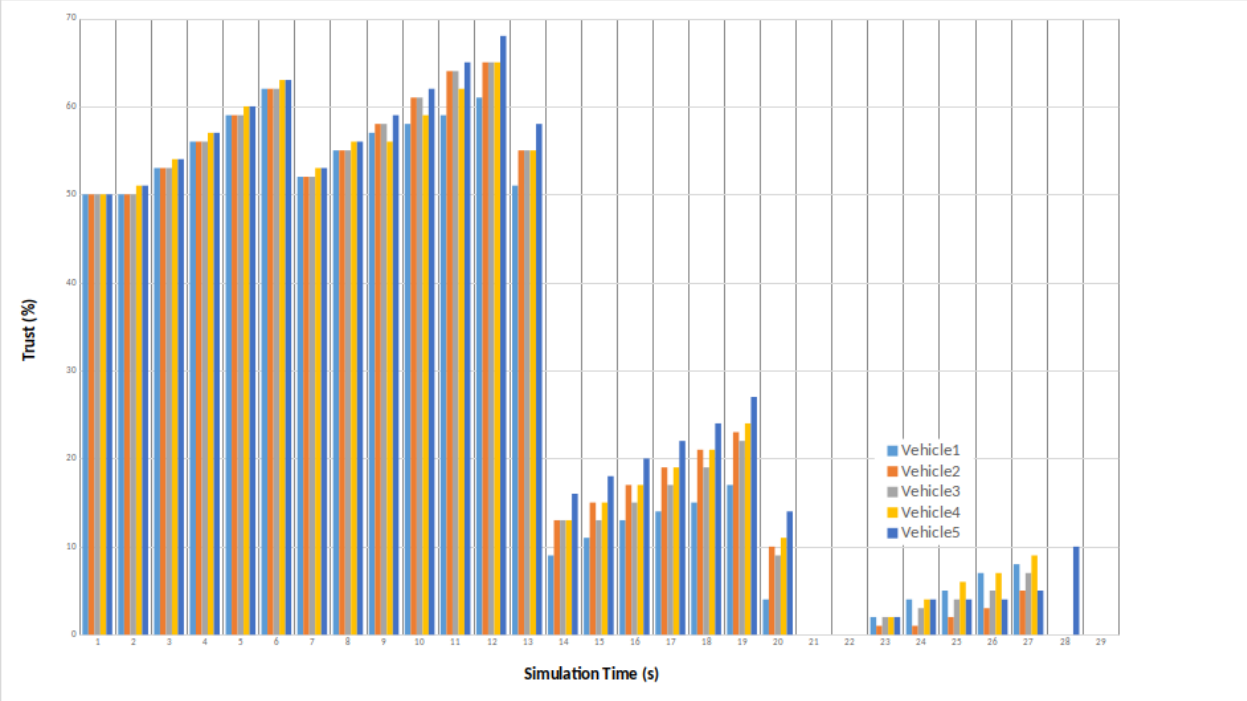

Figure 6.8 Trust value results from a simulation of vehicles that demonstrate progression toward becoming increasingly more malicious

trust of 90%. Vehicle 1 then sends a couple of malicious messages and by the end of the simulation has an overall trust value of 70%. Vehicle 2 and 3 follow a similar pattern where there is an initial increase in trust followed by an anomalous message that decreases both trust values by approximately the same 15%. Vehicle 3 continues to send messages with a few anomalies such that its trust value hovers around 60-70%. Vehicle 2, however, proceeds to primarily send trustworthy messages thus is able to increase it's trust to 97%, nearly a perfect trust value. Vehicle 5 on the other hand primarily sends accurate messages with a small number of complications that leads it to earning a 100% trust value after only 33 messages. Vehicle 4, however, starts off quite negatively, almost entirely maintaining a trust value below 40% up until message 26. At which point vehicle 4 changes paths to sending primarily positive messages and is able to increase its trust value almost to that of the others at a 62%. These results are demonstrated in Figure 6.9.

Figure 6.9 Trust value results from a simulation of vehicle that will randomly injecting malicious data

These results are absolutely sporadic, however, because of the nature of connected vehicle data and how highly dynamic the environment is it is reasonable to assume that there will be instances when the data and sensors report data that does not match the expected behavior that was collected for the training models. It is, however, expected to quite random but generally follow a path upwards, which these results display as all trust values of the vehicles increase to above 60% with two reaching nearly perfect trust.

6.3.7 Use case: Illusion-based Simulation

This use case will be used to show how the system reacts during illusion based messages as described in Section 2.4.3, where a malicious vehicle creates the illusion of an incident by imitating

the driving statistics of said incident (described in Table 6.1). These use cases are critical to the evaluation of the consensus mechanism implemented described in 5.3.1. Each of these cases will involve one illusionist with the other four vehicles following behind said illusionist.

The first use case will be used as a baseline for a working consensus mechanism. The illusionist will report illusion based messages simulating a wreck even if one is not present. Then the four vehicles following the illusionist will also report a car collision such that the illusionist's trust is not negatively impacted.

The second use case will again contain the illusionist sending illusion based messages but following *directly* behind it are four vehicles that will not report an accident. This will show how the consensus mechanism is able to retroactively punish the vehicle in the same time unit the messages were received. This is to say that the illusionists trust value increases as it's message was processed first. Then the follower vehicles come in and report that the message was incorrect and the system will then retroactively punish the illusionist, while the four following vehicles all gain trust.

The third use case for illusion based messages involves the demonstration of the *delta* described in 5.3.1. Again there is one illusionist creating the illusion of incidents when they do not exist, with four other vehicles following behind. However, in this case, the four vehicles following behind are approximately five time units back. This means the illusionist will not be disproved for a few time units. When the follower vehicles arrive to the location of the reported incident the consensus mechanism will be activated, and the trust of the vehicle should decrease.

6.3.7.1 Analysis of a majority agreement consensus simulation

With the nature of this simulation being the base case for the further Illusion based threats, trusts here model use good vehicle's use case presented in section 6.3.1, where there is a constant increase to the point all vehicles are able to reach 100%. The simulation in this case demonstrates

that the consensus mechanism was successfully able to agree with the illusionist such that all trust values increase without any retroactive punishments.

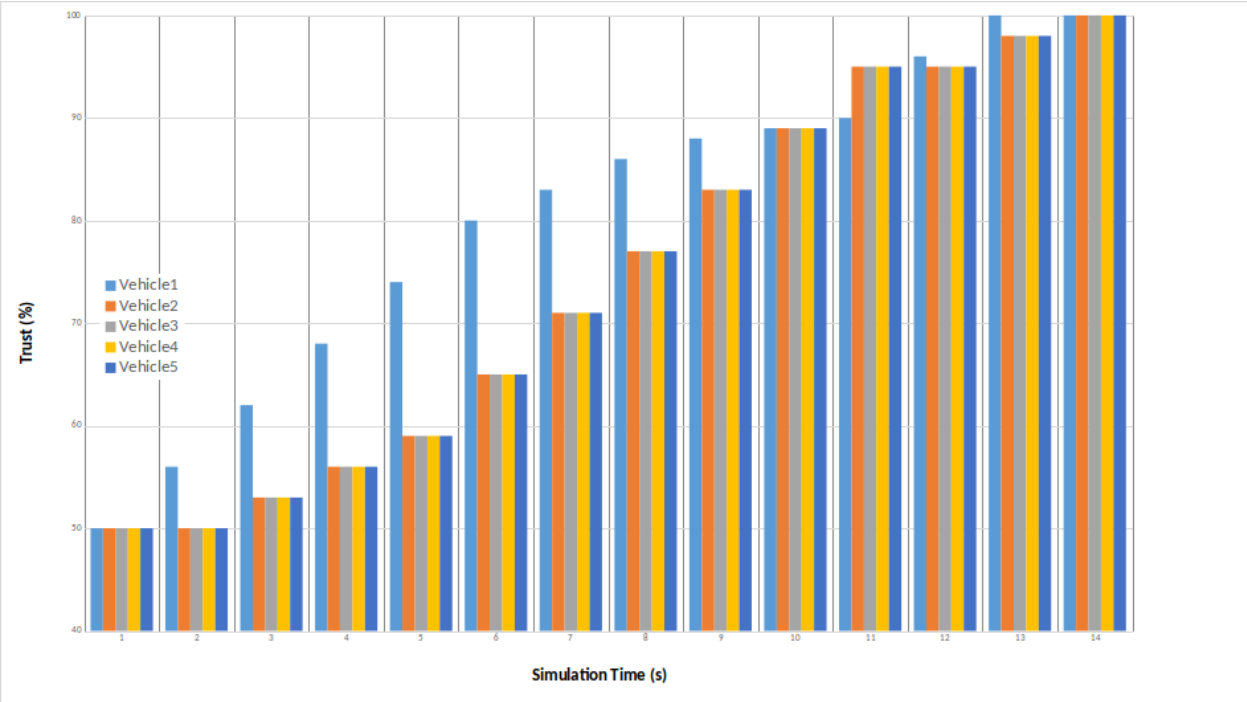

Figure 6.10 Trust value results from a simulation of vehicles that demonstrate a majority consensus

6.3.7.2 Analysis of majority disagreement simulation

As discussed this use case is used to demonstrate the ability to punish illusionists based upon a consensus that the incident reported did not occur. It can be seen that vehicle 1 at time 1 has a trust value of 25%. This does not match the results from use case 2, because consensus mechanism relies on retroactively punishing vehicles. Meaning that at time 1, all data from time 0 has been processes such that vehicle 1's trust was initially increased, but then was quickly disproved by the follower vehicles such that vehicle 1 was then retroactively punished. Further, the

high trustworthiness ratio that vehicle 1 has (as its first message was not an anomaly) the trust value does not instantly drop to 0%. Following this initial decrease, the illusionist proceeds to send more illusion based messages that are disproved by the follower vehicles. The following vehicles are able to steadily increase their trust while the illusionist is effectively removed from the system.

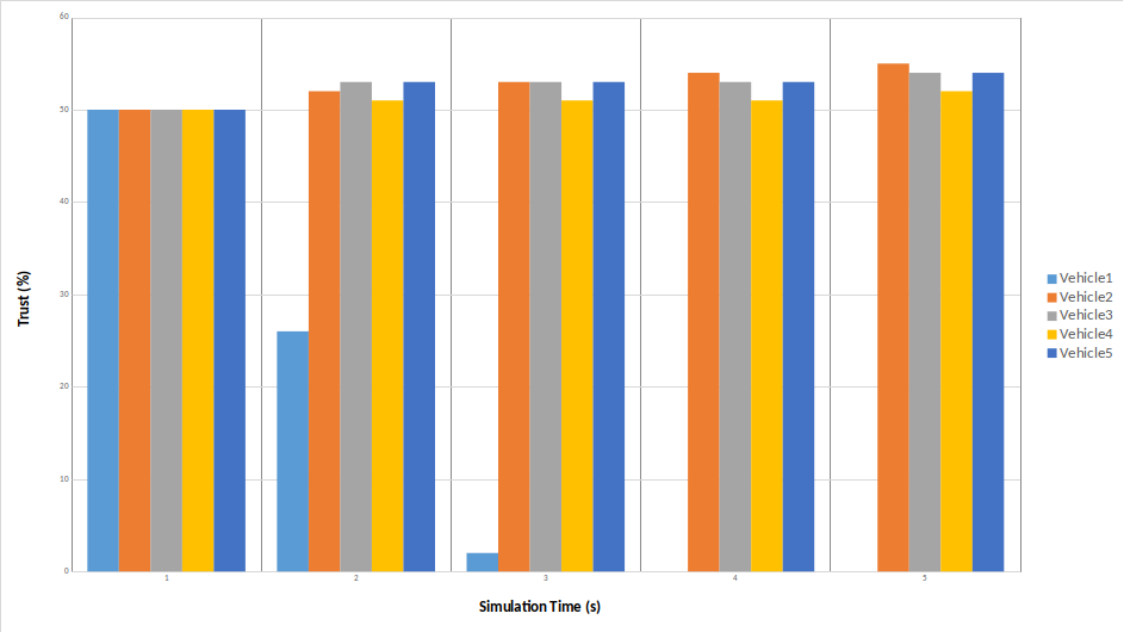

Figure 6.11 Trust value results from a simulation where a leading vehicle is against the majority consensus

6.3.7.3 Analysis of delayed majority disagreement simulation

This use case is an extension where there is still one illusionist and four follower vehicles. Except this time the following vehicles are 5 time units behind. Thus giving way to further testing of the consensus mechanism and ability to retroactively punish vehicles. Figure 6.12 clearly displays that vehicle 1, the illusionist, is driving along the road sending illusions that fools the system into increasing the trust value of the vehicle. However, when the follower vehicles arrive to the location that the illusionist reported an incident, they do not detect that incident that was reported

and is flagged by the consensus mechanism. This enables retroactively punishment the illusionist for the vehicle that was sent at time 0 because it was proven that there was no incident. The results clearly show this occurs at time 5 where vehicle 1's trust drops nearly 70% in just 2 retroactive punishments. These results clearly show that the implementation of the *delta* in the consensus mechanism works as expected. These results are demonstrated in Figure 6.12.

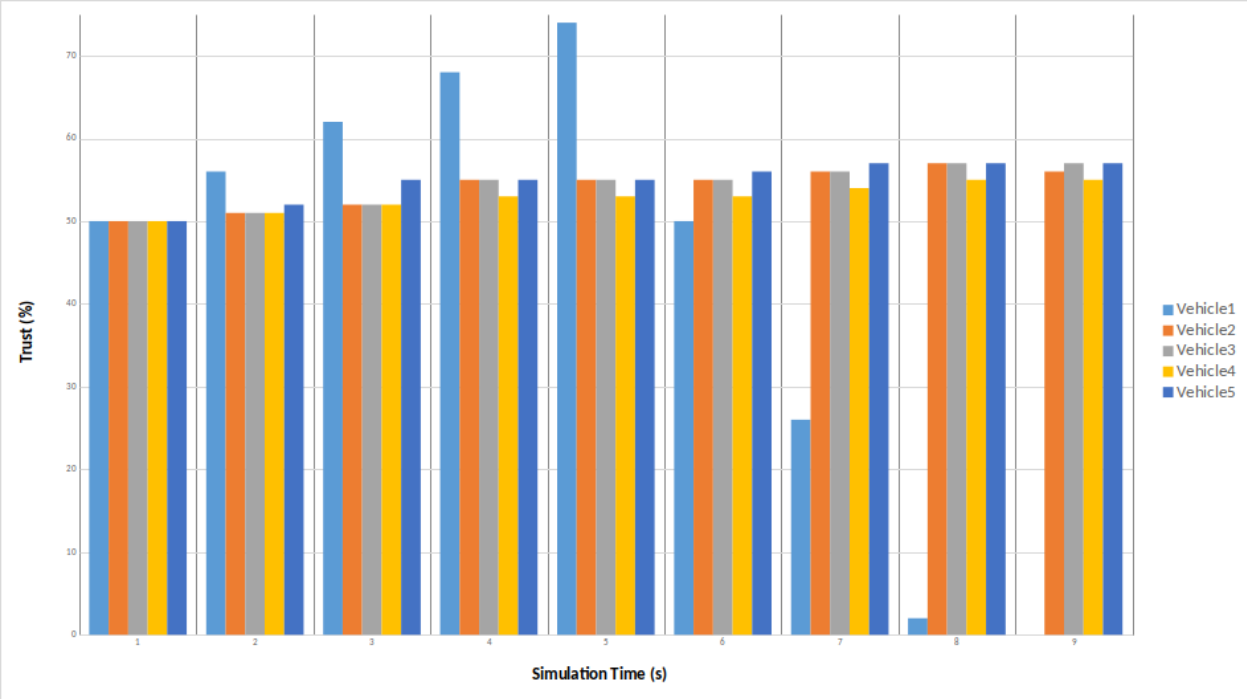

Figure 6.12 Trust value results from a simulation where there is a delay in the disagreement on the majority consensus

CHAPTER 7

Conclusion

We offer conclusions and mention future work here.

7.1 Closing Thoughts

This work has shown that: the collection of a vehicle's driving statistics allows for the creation of a behavioral model that describes how vehicles in a given geographical location behave; using a two-tier machine learning model it is possible to detect malicious behavior in real-time and with over 90% accuracy; and through this detection process, a behavioral value can be calculated using the vehicle's trustworthiness ratio such that this behavioral value can be used as an update to the vehicle' trust value to further aid traditional trust management approaches.

This design sought to build onto the weaknesses that current trust management systems suffer from, these weaknesses include the punishment of malicious actors in real time as well as mitigating colluding attacks which seek to exploit traditional consensus mechanisms. In current trust management approaches, when a majority of vehicles in the network are malicious and injecting inaccurate data into the system, then there is no method that validates the data that these vehicles are sending but instead solely rely on the consensus of data to update the trust values.

Furthermore, the behavioral based trust management approach was designed which implemented three distinct contributions to trust management: the construction of a behavioral model, a behavioral pattern identification process, and a behavioral value. Through detailed data collection a behavioral model can be designed for a localized area which accurately represents the

standard behavior of vehicles within the area. Then a behavioral pattern identification system was constructed which utilized two forms of machine learning, anomaly detection and classification algorithms which when new messages from vehicles were input, the data would be extracted and compared to the behavioral model that was previously created. Through this analysis of data any vehicle whose driving statistics do not match the standard or expected behavior will be classified as an anomaly, as an additional layer of protection these anomalies would then be processed by the classification algorithm which would determine if the driving statistics correspond to the vehicle's message (i.e., a false positive), or if the message is a true anomaly. Lastly after the behavioral pattern identification system, a behavioral value would be calculated. This behavioral value is based upon the trustworthiness ratio, which is defined as the number of anomalies per the total messages the vehicle has transmitted, as well as the current messages rank. These two components allows for vehicle's who have a low trustworthiness ratio quickly be able to lose trust while simultaneously making it harder to gain trust. While also making vehicle's who have a high trustworthiness ratio to not be critically punished on the first inaccurate message it has sent.

This behavioral based trust management trust approach has been shown to detect and mitigate threats which other trust management approach suffer from such as: breakout fraud, selective behavior, illusion-based attacks, and colluding attacks. Results show that these threats were mitigated in real-time while not overly punishing trustworthy vehicles who make mistakes.

7.2 Future Work

Future work for this project can be found below, Section 7.2.1 discusses how real world connected vehicle data would have benefited this work, Section 7.2.2 described how a blockchain implementation would protect data, followed by Section 7.2.3 where a potential accuracy rating would also be applied to trust modification.

7.2.1 Real World Implementation

One of the shortcomings of this work is the simulation of traffic. SUMO was satisfactory and allowed for successful modeling of traffic and connected vehicles, but using a software based application that simulated traffic and imitate collisions can certainly be improved if it were to use real world connected vehicle data. NHTSA stated that the stated the sensors and communication devices could be added to vehicles for as little as $350, so with potential funding it would be possible to convert a vehicle into a connected vehicle and obtain production grade connected vehicle data. Furthermore testing this implementation on true edge devices instead of a desktop would provide more insightful results for the performance evaluations of the different machine learning models and how this approach would behave under those circumstances.

7.2.2 Secure Decentralized Trust Database

Trust management approaches are able to mitigate threats a vehicle is behaving maliciously for a period of time. However if a malicious actor were to hack the device storing the trust values of all vehicles and modify them to all having a 100% then the mitigation of threats is completely irrelevant. In our previous works we have demonstrated that connected vehicles and smart cities are capable of generating and maintaining a lightweight blockchain [22]. A blockchain implementation would aid in the decentralized approach, while providing key features such as tamper-proofing and consistency of the data. I believe there would also be potential for storage of the behavioral model on a blockchain which could have potential for data analysis upon the behavior of the vehicle across its lifetime instead of just a 24 hour period.

7.2.3 Behavioral Value and Trust Modification Evaluation

In current trust management approach as well as this work, a message is evaluated as either trustworthy or not. With this work, an additional security layer has been added that is able to evaluate the data that is being transmitted. By evaluating this data the behavioral value formula for modifying doesn't have to be a pure true or false. The classification machine learning algorithm is able to classify messages which were anomalies that would enable for an accuracy rating of anomalies to be applied. That is, vehicles that are reporting a car collision but are potentially driving too fast will be evaluated as an anomaly and thus punished for lying about a car crash, instead the classification algorithm would see that the vehicle's message's rank was off by a rank of one or two, and thus not be severely punished but instead have their trust decreased at a lesser amount. Doubts remain regarding this strategy, however, because it could potentially not mitigates threats to the system fast enough to where those malicious vehicles would be able to send several messages (which could negatively impact the system).

REFERENCES

[1] Scikit, "Comparing anomaly detection algorithms for outlier detection on toy datasets." [Online]. Available: https://scikit-learn.org/stable/auto_examples/miscellaneous/plot_anomaly_comparison.html#sphx-glr-auto-examples-miscellaneous-plot-anomaly-comparison-py

[2] ——, "Classifier comparison." [Online]. Available: https://scikit-learn.org/stable/auto_examples/classification/plot_classifier_comparison.html#sphx-glr-auto-examples-classification-plot-classifier-comparison-py

[3] S. S. Manvi and S. Tangade, "A survey on authentication schemes in vanets for secured communication," *Vehicular Communications*, vol. 9, pp. 19 – 30, 2017. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S2214209616300018

[4] A. Vehicles, "Self-driving vehicles enhanced legislation," http://www.ncsl.org/research/transportation/autonomous-vehicles-self-driving-vehicles-enacted-legislation.aspx, accessed on: June 2017.

[5] R. Khatoun and S. Zeadally, "Cybersecurity and privacy solutions in smart cities," *IEEE Communications Magazine*, vol. 55, no. 3, pp. 51–59, 2017.

[6] C. A. Kerrache, C. T. Calafate, J.-C. Cano, N. Lagraa, and P. Manzoni, "Trust management for vehicular networks: An adversary-oriented overview," *IEEE Access*, vol. 4, pp. 9293–9307, 2016.

[7] M. Blaze, J. Feigenbaum, and J. Lacy, "Decentralized trust management," in *Proceedings 1996 IEEE Symposium on Security and Privacy*. IEEE, 1996, pp. 164–173.

[8] L. Xiong and L. Liu, "Building trust in decentralized peer-to-peer electronic communities," in *Fifth International Conference on Electronic Commerce Research (ICECR-5)*, 2002.

[9] ——, "Peertrust: Supporting reputation-based trust for peer-to-peer electronic communities," *IEEE transactions on Knowledge and Data Engineering*, vol. 16, no. 7, pp. 843–857, 2004.

[10] W. Li, H. Song, and F. Zeng, "Policy-based secure and trustworthy sensing for internet of things in smart cities," *IEEE Internet of Things Journal*, vol. 5, no. 2, pp. 716–723, 2017.

[11] F. Kandah, A. Altarawneh, B. Huber, A. Skjellum, and S. Medury, "A human-understandable, behavior-based trust management approach for iot/cps at scale," 2020.

[12] NHTSA, "Vehicle-to-vehicle communication technology." [Online]. Available: https://www.nhtsa.gov/sites/nhtsa.dot.gov/files/documents/v2v_fact_sheet_101414_v2a.pdf

[13] L. Atzori, A. Iera, and G. Morabito, "The internet of things: A survey," *Computer networks*, vol. 54, no. 15, pp. 2787–2805, 2010.

[14] A. Zanella, N. Bui, A. Castellani, L. Vangelista, and M. Zorzi, "Internet of things for smart cities," *IEEE Internet of Things journal*, vol. 1, no. 1, pp. 22–32, 2014.

[15] K. Leuth, "State of the iot 2018." [Online]. Available: https://iot-analytics.com/state-of-the-iot-update-q1-q2-2018-number-of-iot-devices-now-7b/

[16] Thales, "What is a smart city?" [Online]. Available: https://www.thalesgroup.com/en/markets/digital-identity-and-security/iot/inspired/smart-cities

[17] D. Washburn, U. Sindhu, S. Balaouras, R. A. Dines, N. Hayes, and L. E. Nelson, "Helping CIOs understand "smart city" initiatives," *Growth*, vol. 17, no. 2, pp. 1–17, 2009.

[18] A. Kanungo, A. Sharma, and C. Singla, "Smart traffic lights switching and traffic density calculation using video processing," in *2014 Recent Advances in Engineering and Computational Sciences (RAECS)*. IEEE, 2014, pp. 1–6.

[19] S.-I. Sou and O. K. Tonguz, "Enhancing vanet connectivity through roadside units on highways," *IEEE transactions on vehicular technology*, vol. 60, no. 8, pp. 3586–3602, 2011.

[20] X. Yang, L. Liu, N. H. Vaidya, and F. Zhao, "A vehicle-to-vehicle communication protocol for cooperative collision warning," in *The First Annual International Conference on Mobile and Ubiquitous Systems: Networking and Services, 2004. MOBIQUITOUS 2004.* IEEE, 2004, pp. 114–123.

[21] J. Zou, B. Ye, L. Qu, Y. Wang, M. A. Orgun, and L. Li, "A proof-of-trust consensus protocol for enhancing accountability in crowdsourcing services," *IEEE Transactions on Services Computing*, vol. 12, no. 3, pp. 429–445, 2018.

[22] F. Kandah, B. Huber, A. Altarawneh, S. Medury, and A. Skjellum, "Blast: Blockchain-based trust management in smart cities and connected vehicles setup," in *2019 IEEE High Performance Extreme Computing Conference (HPEC)*. IEEE, 2019, pp. 1–7.

[23] Merriam-Webster, "Machine learning." [Online]. Available: https://www.merriam-webster.com/words-at-play/what-does-machine-learning-mean

[24] B. C. Love, "Comparing supervised and unsupervised category learning," *Psychonomic bulletin & review*, vol. 9, no. 4, pp. 829–835, 2002.

[25] J. C. Ang, A. Mirzal, H. Haron, and H. N. A. Hamed, "Supervised, unsupervised, and semi-supervised feature selection: a review on gene selection," *IEEE/ACM transactions on computational biology and bioinformatics*, vol. 13, no. 5, pp. 971–989, 2015.

[26] W. Liu, Z. Wang, X. Liu, N. Zeng, Y. Liu, and F. E. Alsaadi, "A survey of deep neural network architectures and their applications," *Neurocomputing*, vol. 234, pp. 11–26, 2017.

[27] T. Lane and C. E. Brodley, "An application of machine learning to anomaly detection," in *Proceedings of the 20th National Information Systems Security Conference*, vol. 377. Baltimore, USA, 1997, pp. 366–380.

[28] C.-W. Ten, J. Hong, and C.-C. Liu, "Anomaly detection for cybersecurity of the substations," *IEEE Transactions on Smart Grid*, vol. 2, no. 4, pp. 865–873, 2011.

[29] T. Shon and J. Moon, "A hybrid machine learning approach to network anomaly detection," *Information Sciences*, vol. 177, no. 18, pp. 3799–3821, 2007.

[30] F. Pereira, T. Mitchell, and M. Botvinick, "Machine learning classifiers and fmri: a tutorial overview," *Neuroimage*, vol. 45, no. 1, pp. S199–S209, 2009.

[31] L. Cheng, B. E. Henty, D. D. Stancil, F. Bai, and P. Mudalige, "Mobile vehicle-to-vehicle narrow-band channel measurement and characterization of the 5.9 ghz dedicated short range communication (dsrc) frequency band," *IEEE Journal on Selected Areas in Communications*, vol. 25, no. 8, pp. 1501–1516, 2007.

[32] A. Tajeddine, A. Kayssi, and A. Chehab, "A privacy-preserving trust model for vanets," in *2010 10th IEEE International Conference on Computer and Information Technology*. IEEE, 2010, pp. 832–837.

[33] Z. Yang, K. Yang, L. Lei, K. Zheng, and V. C. Leung, "Blockchain-based decentralized trust management in vehicular networks," *IEEE Internet of Things Journal*, vol. 6, no. 2, pp. 1495–1505, 2018.

[34] J. Jiang, G. Han, F. Wang, L. Shu, and M. Guizani, "An efficient distributed trust model for wireless sensor networks," *IEEE transactions on parallel and distributed systems*, vol. 26, no. 5, pp. 1228–1237, 2014.

[35] R. Feng, X. Xu, X. Zhou, and J. Wan, "A trust evaluation algorithm for wireless sensor networks based on node behaviors and ds evidence theory," *Sensors*, vol. 11, no. 2, pp. 1345–1360, 2011.

[36] G. Zhan, W. Shi, and J. Deng, "Design and implementation of tarf: A trust-aware routing framework for wsns," *IEEE Transactions on dependable and secure computing*, vol. 9, no. 2, pp. 184–197, 2011.

[37] I. Butun, S. D. Morgera, and R. Sankar, "A survey of intrusion detection systems in wireless sensor networks," *IEEE communications surveys & tutorials*, vol. 16, no. 1, pp. 266–282, 2013.

[38] G. S. Khekare and A. V. Sakhare, "A smart city framework for intelligent traffic system using vanet," in *2013 International Mutli-Conference on Automation, Computing, Communication, Control and Compressed Sensing (iMac4s)*. IEEE, 2013, pp. 302–305.

[39] S. Jarvenpaa, T. R. Shaw, and D. Sandy Staples, "Toward contextualized theories of trust: The role of trust in global virtual teams," *Information Systems Research*, vol. 15, pp. 250–267, 09 2004.

[40] J. Ali, T. Ali, Y. Alsaawy, A. S. Khalid, and S. Musa, "Blockchain-based Smart-IoT Trust Zone Measurement Architecture," in *Proceedings of the International Conference on Omni-Layer Intelligent Systems*, ser. COINS '19. New York, NY, USA: ACM, 2019, pp. 152–157. [Online]. Available: http://doi.acm.org/10.1145/3312614.3312646

[41] I. Alrashdi, A. Alqazzaz, E. Aloufi, R. Alharthi, M. Zohdy, and H. Ming, "Ad-iot: Anomaly detection of iot cyberattacks in smart city using machine learning," in *2019 IEEE 9th Annual Computing and Communication Workshop and Conference (CCWC)*. IEEE, 2019, pp. 0305–0310.

[42] V. Garcia-Font, C. Garrigues, and H. Rifà-Pous, "A comparative study of anomaly detection techniques for smart city wireless sensor networks," *sensors*, vol. 16, no. 6, p. 868, 2016.

[43] H. H. Pajouh, G. Dastghaibyfard, and S. Hashemi, "Two-tier network anomaly detection model: a machine learning approach," *Journal of Intelligent Information Systems*, vol. 48, no. 1, pp. 61–74, 2017.

[44] R. De Maesschalck, D. Jouan-Rimbaud, and D. L. Massart, "The mahalanobis distance," *Chemometrics and intelligent laboratory systems*, vol. 50, no. 1, pp. 1–18, 2000.

[45] J. Lee, B. Kang, and S.-H. Kang, "Integrating independent component analysis and local outlier factor for plant-wide process monitoring," *Journal of Process Control*, vol. 21, no. 7, pp. 1011–1021, 2011.

[46] S. C. Johnson, "Hierarchical clustering schemes," *Psychometrika*, vol. 32, no. 3, pp. 241–254, 1967.

[47] W. S. Noble, "What is a support vector machine?" *Nature biotechnology*, vol. 24, no. 12, pp. 1565–1567, 2006.

[48] J. Chin, V. Callaghan, and I. Lam, "Understanding and personalising smart city services using machine learning, the internet-of-things and big data," in *2017 IEEE 26th International Symposium on Industrial Electronics (ISIE)*. IEEE, 2017, pp. 2050–2055.

[49] T. S. Brisimi, C. G. Cassandras, C. Osgood, I. C. Paschalidis, and Y. Zhang, "Sensing and classifying roadway obstacles in smart cities: The street bump system," *IEEE Access*, vol. 4, pp. 1301–1312, 2016.

[50] H. Hasrouny, C. Bassil, A. E. Samhat, and A. Laouiti, "Group-based authentication in v2v communications," in *2015 Fifth International Conference on Digital Information and Communication Technology and its Applications (DICTAP)*. IEEE, 2015, pp. 173–177.

[51] R. Van Der Heijden, "Security architectures in v2v and v2i communication," in *Proc. 20th Student Conf. IT*, 2010, pp. 1–10.

[52] D. Krajzewicz, J. Erdmann, M. Behrisch, and L. Bieker, "Recent development and applications of sumo-simulation of urban mobility," *International journal on advances in systems and measurements*, vol. 5, no. 3&4, 2012.

[53] D. Krajzewicz, "Traffic simulation with sumo–simulation of urban mobility," in *Fundamentals of traffic simulation*. Springer, 2010, pp. 269–293.

[54] M. Haklay and P. Weber, "Openstreetmap: User-generated street maps," *IEEE Pervasive Computing*, vol. 7, no. 4, pp. 12–18, 2008.

[55] A. R. Schmidt, F. Waas, M. L. Kersten, D. Florescu, I. Manolescu, M. J. Carey, R. Busse *et al.*, "The xml benchmark project," 2001.

[56] M. Behrisch and R. Hilbrich, "Simulation/output." [Online]. Available: https://sumo.dlr.de/docs/Simulation/Output.html

[57] D. M. Gavrila and V. Philomin, "Real-time object detection for" smart" vehicles," in *Proceedings of the Seventh IEEE International Conference on Computer Vision*, vol. 1. IEEE, 1999, pp. 87–93.

[58] L. Sumi and V. Ranga, "Intelligent traffic management system for prioritizing emergency vehicles in a smart city," *International Journal of Engineering*, vol. 31, no. 2, pp. 278–283, 2018.

[59] M. Behrisch and R. Hilbrich, "Simulation/routing." [Online]. Available: https://sumo.dlr.de/docs/Simulation/Routing.html

[60] P. development team, "Pandas user guide." [Online]. Available: https://pandas.pydata.org/docs/user_guide/index.html

[61] J. Coleman, F. Kandah, and B. Huber, "Behavioral model anomaly detection in automatic identification systems (ais)," in *2020 10th Annual Computing and Communication Workshop and Conference (CCWC)*. IEEE, 2020, pp. 0481–0487.

VITA

Brennan Huber, born in Nassau Bay, Texas, grew up in Paducah, Kentucky, and graduated high school in Greenbrier, Tennessee. Brennan graduated college from the University of Tennessee Chattanooga with Bachelors of Science in Computer Science: Scientific Application. Brennan continued his education by pursuing a Master of Science in Computer Science: Cybersecurity. His primary interests include cybersecurity, trust management systems, and applications of blockchain. While pursing the Master's, Brennan was a full time employee at BlueCross BlueShield of Tennessee as an Applications Developer. Brennan plans to advance his education further through a Ph.D. program in Computational Science at the University of Tennessee Chattanooga.