Trustcito: A trust management system for distributed trust in V2X

by Yu Sen Chiu

Bachelor's degree in Computer Science, 06 2022, National Defense University

A Thesis submitted to

The Faculty of
The School of Engineering and Applied Science
of The George Washington University
in partial satisfaction of the requirements
for the degree of Master of Cybersecurity in Computer Science

May 19, 2024

Thesis directed by

Sibin Mohan
Associate Professor of Computer Science

The School of Engineering and Applied Science of The George Washington University certifies that Yu Sen Chiu has passed the Final Examination for the degree of Master of Cybersecurity in Computer Science as of April 12, 2024. This is the final and approved form of the Thesis.

Trustcito: A trust management system for distributed trust in V2X

Yu Sen Chiu

Thesis Research Committee:

Sibin Mohan, Associate Professor of Computer Science, Thesis Director

Timothy Wood, Associate Professor, Committee Member

Arkady Yerukhimovich, Assistant Professor, Committee Member

Acknowledgments

I would like to express my appreciate my thesis advisor Sibin Mohan for his invaluable guide and feedback. I cannot finish my thesis without his patience and expertise. Every time I asked questions to him, he responds to me as soon as possible.

I am also grateful to the S Lab members who give me a lot of help during my thesis. They help me to get familiar with V2X background knowledge and Simulation usage. I also want to thank for my friends Ting Hung Chiu who gave me a lot of suggestions and encouraged me to accomplish my goals.

Ultimately, I would like to thank for my parents. They always support and inspire me to move forward.

Abstract

Trustcito: A trust management system for distributed trust in V2X

Intelligent Transportation Systems (ITS) play a crucial role in realizing sustainable transportation by addressing traffic congestion and ensuring safe travel experiences. Among the key components of ITS, V2X communication facilitates the exchange of information between vehicles and ITS entities. However, despite its potential benefits, the open wireless nature of V2X communication renders it vulnerable to attacks. In response, many researchers has provided different defense schemes. These schemes often rely on direct interactions to identify malicious vehicles. However, direct experiences of target vehicles are often unavailable due to network disruptions or encounters with new users makes these defense out of work. In response, I introduce the *Trustcito* framework, a recommendation-based trust management system that can be used in every vehicle. This framework continuously monitors potential malicious vehicles by requesting recommendations from other entities and then evaluates them against a predefined threshold. my analysis demonstrates that indirect trust can serve as a valuable complement to direct trust within the proposed model. The performance of proposed model is evaluated by Query-Hit Rate and precision recall metrics. For accuracy, I achieve a high precision of above 70% in every map. Regarding the query-hit rate, if there are many nodes and edges, I can achieve a query-hit rate of over 70%.

Table of Contents

List of Figures

List of Tables

List of Abbreviations

V2X Vehicle to Everything

ITS Intelligent Transportation System

V2I Vehicle to Infrastructure

V2N Vehicle to Network

V2V Vehicle to Vehicle

V2P Vehicle to Pedestrian

V2D Vehicle to Device

BSM Basic Safety Message

MANET Mobile ad hoc networks

VANET Vehicular ad hoc networks

Dos Denial of Service

DDos Distributed Denial of Service

TTL Time To Live

P2P Peer to Peer

Preface

Intelligent Transportation Systems (ITS) are revolutionizing transportation with their promise of efficiency and safety. However, the open nature of Vehicle-to-Everything (V2X) communication introduces cybersecurity challenges. Traditional defense schemes often fall short in dynamic vehicular networks.

This thesis introduces the *Trustcito* framework, a recommendation-based trust management system for V2X security. By leveraging indirect trust through recommendations, *Trustcito* enhances security resilience. Evaluation metrics like Query-Hit Rate and precision-recall demonstrate its effectiveness.

This work contributes to the evolution of Vehicle to Everything (V2X) security, offering practical solutions for a safer and more reliable transportation future.

Chapter 1: Introduction

As the Internet developed, many connected systems showed up in this generation. Due to traffic congestion, vehicle accidents, and traffic problems, an Intelligent Transportation System (ITS) has been developed [16] [47]. ITS can make an efficient transportation system by connecting all nodes for each other and communicating to each node with certain protocols. V2X is based on ITS. It includes protocols Vehicle to Infrastructure (V2I), Vehicle to Network (V2N), Vehicle to Vehicle (V2V), Vehicle to Pedestrian (V2P), and Vehicle to Device (V2D) [38]. By adopting V2X technology, every single car can be connected to each other as well as roadside units. As a result, each vehicle can not only get traffic information from other vehicles and roadside units, but also receive alerts of incidents or unexpected accidents [45]. For example, V2V communication allows vehicles to exchange information with nearby vehicles, that includes location, speed, acceleration, and direction. This enables advanced safety features like collision avoidance and safety at intersections. V2I enables vehicles to communicate with infrastructure elements such as traffic lights, road signs and toll booths. This allows for optimized traffic flow, improved navigation and enhanced safety through real-time information exchange. V2P allows vehicles to communicate with pedestrians via their phones or personal devices. This can provide warnings to pedestrians about approaching vehicles, enhancing safety for both drivers and pedestrians.

However, there are many threats in the vehicular network [26]. For example: Sybil-attack, Denial of Service...etc. The adversary model will be detailed in chapter 3.These threats can be split into 2 subsets: inter-

nal and external. The internal attackers means those attackers who have already authenticated and joined into the network while external attackers means non-authenticated attackers [3] [5].Internal attackers can be more dangerous since they have already passed the authentication process. Once authenticated, they can send malicious messages to others from the network. For instance, attackers could convince individuals to detour to hazardous locations. Without a way to prevent this behavior, it could result in significant damage.

To deal with threats, many researchers have focused on how to deny access to these attackers in vehicular networks. Some provide Crypto schemes such as PKI-infrastructure to defend against malicious users [19] [21]. However, these mechanisms can only be used against external attackers since internal attackers has already passed the authenticated challenges. To address the internal attacker problems, there are 3 main proposed solutions [29]. One is data-centric system [57] [63] [61] another one is node-based system [33] [32] [4] and the other is hybrid.[67] [14]. Data-centric systems: these kind of systems validate the accuracy and plausibility of a packet's content. The packet format used in V2X is the Basic Safety Message (BSM) [34] [2]. There is a lot of information in the BSM packet such as vehicle status, data, and vehicle behavior. The node-based method is basically based on trust established by each vehicles on its own. Every entity will generate a trust score after interacting with others. Some research combines trust and data consistency to form a hybrid system.

There are 2 different ways to calculate trust in node-based schemes. One is direct trust [28] [7] [27], and the other is indirect trust [37] [38] [6]. In direct trust, vehicles interact with other vehicles so that it can calculate the trust score. Indirect trust is based on recommendations from the others.

In this thesis, I make a hypothesis that people can use trust that combines direct trust and indirect trust to defend against malicious users in vehicular systems. However, due to the limitations direct trust including network problem, indirect trust can also be a helpful indicator to determine trust relations. I need a recommendation-based system to develop a trust score. To prove that, I provide a node-based scheme named *Trustcito*. This system is based on node-based system which used into P2P system named "VectorTrust" [69]. I will discuss VectorTrust in chapter 2 and discuss my system model in chapter 4. I also add a trust decay feature in my model since I believe that trust cannot be 100% transit in real-world case.

I have the following problem statements.

1. How can people generate and use indirect trust scores?
2. If direct trust is unavailable, how can vehicles trust each other?
3. What are the challenges faced while calculating indirect trust?
4. Why do we need V2X solutions instead of existing solutions on peer to peer trust?

I designed my Trustcito framework to address these problems. There are many 1 to 1 trust management systems in the world. However, if the vehicle has not directly interacted with another car, how can others trust it? As a result, I develop a recommendation-based trust management system to solve this problem for one to many targets.

In this paper, I make the following of contributions:

1. Address the limitation of direct trust in situations where it is unavailable
2. Distinguish between benign vehicles and malicious vehicles.

3. Address the use of a decentralized management system.

The paper is organized as follows. Chapter 2 provides some overview of related work. Chapter 3 describes some common attacks that happen in V2X and I will discuss the threat model. Chapter 4 shows my system model. Chapter 5 explains some normal faults that happen in V2X networks. Chapter 6 discusses my experiments and result with some analysis. Chapter 7, is the conclusion.

Chapter 2: Related Work

There are multiple different trust management schemes that exist. They are catagorized into Peer to Peer (P2P), Mobile ad hoc networks (MANET) and Vehicular ad hoc networks (VANET).

Zhao et al. [69] present the "VectorTrust" that used a product scheme to chain every node in trust transit. They used a local trust table to store every direct node 's trust with neighbors and get an initial trust score. Then they used different stages to transit the trust across the neighbors to the target node. In their scheme, they forwarded every node's local trust table to the next hop (i.e., neighbor). After receiving a neighbor's information, every node will re-calculate the trust path and score to every node. If the new result is bigger than the original trust score, they will replace the new result in their local trust table. In *Trustcito*, I add a decay function to allow the trust to decay based on path length.

Aifarez et al. [1] also use path product to calculate the trust from source to target. They used a distributed number to determine trust level. They used "-1" as distrust, while 0-4 is a level to increase the trust. Then they multiplied every trust score in the path from source to target. They use average to determine the final trust score if there exists multiple paths from source to target. The paper is used in the P2P system, too. In *Trustcito*,I used max value instead of average value as my indirect trust score. The reasonis we believe that people will trust closer friend's recommendation more than normal friend's recommendation.

Alnasser et al. [6] provide a scheme that calculates indirect trust with product scheme. Then based on different situations, they used a different

combination of direct trust and indirect trust. They used a credential based scheme for recommendations. If the recommender 's trust is over threshold, the recommendation will be fully accepted. If its trust is between low and high threshold, then the recommendation will be partially accepted. Otherwise, the recommendation will be dropped. In *Trustcito*, I accept all recommendations then pick the best among them.

Banerjee et al. [9] used a windows-based scheme to calculate the trust. Every node will get an assigned sized window to store the trust. They used 0 to represent distrust and 1 to represent trust. Then every node in the scenario will contain a credit rating that means how much the node influence the other node's recommendation. Finally, they sum all bits in a window and generate a final trust score. In *Trustcito*, I used a window-based trust table to store the trust.

Xin et al. [35] provide a trust scheme that also used product of trust path in a MANET. They used a packet forwarding rate to calculate the node's trust score. They separate trust level from 1 to 4. If trust score is low ie. level 1, the node will automatically drop the packet from this low level node. They assigned 0.75 initially for all nodes that have not interacted with each other. After interacting with each other, every node will increase or decrease the trust node to each neighbor. Then, they used a routing protocol to transmit data and trust. After discovering the path from source to destination, they also multiplied the whole path trust value. In *Trustcito*, I assign a trust score of 0.5 initially for experiments because it represents the midpoint of the trust scale. I don't want to assume that all vehicles are good vehicles, so a trust score of 0.75 is not suitable for my scenario.

Sanjay et al. [14] use their trust scheme in VANET . There are 4 phases in their model including "Neighbor Discovery", "Data Dispatching", "Decision

Making and Trust Updating" , and "Neighbor Monitoring". After getting a trust score, they make each node monitor each other in order to identify data that is sent by malicious vs a benign node. In *Trustcito*, vehicles directly send recommendation packets to each other.

To address the forth point in our problem statement four, we offer the following explanation: Unlike traditional peer-to-peer networks, V2X operates in dynamic, geographically dispersed settings where real-time communication and decision-making are crucial. For instance, consider an autonomous vehicle navigating through a busy intersection; it must not only trust the recommendations of nearby vehicles to ensure safe passage but also consider dynamic factors such as road conditions, traffic patterns and pedestrian movements. The complexity of V2X systems stems from the multitude of factors that need to be considered.

Furthermore, traditional peer-to-peer (P2P) systems are designed for static networks, whereas V2X systems are designed for dynamic networks. Vehicles can move anywhere, while computers in P2P systems may not relocate frequently. The geography and roadside units may also change over time. When the geography differs from the previous version, the V2X network could be altered. Thus, dealing with V2X problems involves not only considering node connections but also addressing the dynamic network environment.

Chapter 3: Threat Model

There are many attacks in the vehicular network. These attacks can be separated into 5 groups according to attack goals [26].

Table 3.1: Attack table

Attack Goal Goal	**Attacks**
Behavior	Message spoofing attack, Eavesdropping Message Replay Attack, Sybil Attack, Denial of Service (DoS)Attack, Black Hole Attack
Software and Hardware	DoS Attack, Message spoofing attack
Infrastructure	Session Hijacking Attack, Distributed Denial of Service (DDoS) Attack
Privacy	Identity Revealing Attack, Location Tracking
Data Trust	False Message Injection, Hidden Vehicle Attack

3.1 Attacks on behavior

In this kind of attack, I can separate attacks into 2 subsets: one is selfish user and the other one is malicious user.

3.1.1 Selfish Attack

Selfish users may drop packets or not forward received packets to others. They may not carry out harmful behaviors or cause damage to others but they will only do what is benefit for themselves. The common attacks are explained as fellow.

1. **Message spoofing attack**. In message spoofing attack, adversary may send spoofed messages to benign users to misdirect them [44][52]. For example, if the selfish user wants to empty the road of vehicles, they may send a false message to other vehicles that this road is congested then they will also send detour suggestions to benign users. This attack can be defended by checking message integrity.

2. **Eavesdropping**. In this attack, attackers attempt to intercept communication between users in order to get sensitive information. This information can then be used in the future attack. This attack can be defended using Crypto scheme [50] [19].

3.1.2 Malicious Attack

Malicious attackers try to break or cause damage to other vehicles. The common attacks are described below.

1. **Message Replay Attack** Attacker replays previously captured network packets at regular intervals. This attack can lead to unauthorized access or manipulation of data, potentially confusing the authorities. However, it can be defended against using timestamp authentication. [48][44] [11].

2. **Sybil Attack** Attacker generate numbers of fake identities and send message to benign nodes . These fake entities deceive the system into perceiving them as real users. The goal is to affect users' trust relations in decentralized systems. Attackers can combine the Sybil attack with a bad-mouth attack to target benign nodes [17] [48]. This attack can be defended by checking identity for each car.

3. **Denial of Service (Dos) Attack** Attackers will send large numbers of worthless packets to occupy resources, preventing benign cars from transmitting packets to each other. Consequently, this can cause traffic jams or accidents [65][48][44] . This attack can be defended by behavior analysis.

4. **Black Hole Attack** Attackers attract packets sent from other nodes and drop them, disrupting communication. The attacker acts as a

"black-hole," causing legitimate traffic to disappear [41] [20]. This attack can be defended against using a trust scheme. By adjusting trust scores (either decreasing or increasing), I can determine whether to send the packet to neighbors.

3.2 Attacks on Software and Hardware

In this type of attack, attackers focus on software and hardware. The common attacks are shown below.

1. **Denial of Service (Dos) Attack** Attackers will send a large number of worthless packets to occupy resources, preventing benign cars from transmitting packets to each other. Consequently, this can cause traffic jams or accidents [65][48][44]. This attack can be defended by behavior analysis.

2. **Message spoofing attack** In message spoofing attack, adversary may send spoofed messages to benign users to misdirect them [44][52]. For example, if the selfish user wants to empty the road of vehicles, they may send a false message to other vehicles that this road is congested then they will also send detour suggestions to benign users. This attack can be defended by checking message integrity.

3.3 Attacks on Infrastructure

In this attack type, attackers focus on the infrastructure, network. The common attacks are shown below.

1. **Session Hijacking Attack** In V2X networks, session tokens are used to recognize each user's connections and manage their sessions. The

attacker's goal is to compromise the session token, either by stealing or predicting a valid token, to gain unauthorized access [43] [70]. This attack can be defended against by setting up session expiration time. Once the session timeout is reached, users should automatically log out.

2. **Distributed Denial of Service (DDos) Attack** A DDoS attack is an advanced form of a DoS attack where attackers send a massive volume of packets from multiple sources simultaneously to flood the bandwidth of a target network, overwhelming its capacity and causing the entire network to crash. This type of attack is often orchestrated using a botnet, a network of compromised devices controlled by the attacker, which amplifies the impact and makes it harder to mitigate the attack. The difference between Dos and DDos is the source[58][62]. DDos cause more damage than Dos

3.4 Attacks on Privacy

In this attack type, attackers try to steal benign user's privacy. The common attack type are explained below.

1. **Identity Revealing Attack** In this attack, attacker gains unauthorized access to information that can expose the identity of entities involved in the communication network. This could include revealing the identity of vehicles, infrastructure components, or individuals interacting with the system [30][26]. This attack can be defended by using pseudonyms instead of real ID.

2. **Location Tracking** In this attack, attackers track the vehicles' location and path. It could cause severe privacy violations. Furthermore, it can

be used for future attacks. For example, if the president's vehicle is tracked, terrorists may follow his car and attack [40][15]. This attack can also be defended against by using pseudonyms instead of real IDs.

3.5 Attack on Data Trust

In this attack type, attackers will modify or alter the data during transit. The common attack type are explained below.

1. **False Message Injection** Attackers send false messages to benign nodes, misleading them and affecting the overall network. These false messages can contain critical information such as incorrect speed, distance, or acceleration data. This misinformation can lead to potentially dangerous situations on the road or compromise the efficiency and safety of the V2X network [29][55][22]. This attack can be defended by trust-based system.

2. **Hidden Vehicle Attack** In this attack, attackers generate false positions to hide cars and potentially cause accidents. This type of attack manipulates GPS information, leading to what is commonly known as GPS spoofing. Such attacks can result in dangerous situations, including false collision warnings, incorrect traffic flow management, or unauthorized access to secure zones. Defenses against this attack include GPS Spoofing Detection mechanisms [52][64].

Mohammad et. al. categorize a different type of attack happened in V2X. They separate attacks to different risk levels by reproducibility, impact and stealthiness indicators [8].

3.6 My Threat model

In my threat model, I prioritize attacks on the recommendation side over privacy concerns because *Trustcito* is a recommendation-based framework. Attacks on recommendations pose a greater threat to my system. If *Trustcito* cannot provide accurate recommendations, it may lead benign vehicles to mistrust malicious vehicles, resulting in significant damage. For example, malicious vehicles could send detour-to-danger-zone packets to benign vehicles. If the benign vehicles trust these packets, they could end up in danger zones. To address this threat model, I first categorize attacks based on their goals and targets. I consider three types of attacks in my model:

1. Recommendation-Based Attacks
2. Routing-Based Attacks
3. False Message Injections

3.6.1 Recommendation-Based Attack

In recommendation-based attack [68], the attackers may try to send fake recommendations to benign nodes. As a result, the benign nodes will incorrectly trust the malicious vehicles' peers. For example: Good mouth attack. In Good mouth attack, attackers will send false, but high trust score recommendations to the benign node. Once the benign node accepts these recommendations, they will trust these malicious cars, and accept their packets. On the other hand, benign nodes will distrust other benign nodes in Bad mouth Attacks. In such situations, attackers try to slander benign vehicles. As soon as benign vehicles trust this fake information, the benign vehicles will disconnect from other benign vehicles and drop packets sent

from those vehicles[3][39][49]. There is another attack called Sybil attack [17][48], which will create a lot of fake entities to confuse the benign cars.

Another problem is that of the New User problem. Vehicles cannot get any information on new vehicles, so that attacker can easily pretend benign cars mix into the new users and cause damage.

3.6.2 Routing-Based Attack

In Routing-based attacks, the adversary tries to focus on the routing behavior and make every effort to interfere with the connection. Furthermore, they may try to block the connection. The common example is Denial of Service (Dos) attack [65][48][44]. Attackers send many and many worthless packets to choke the whole bandwidth, so that vehicles cannot connect to each other.

The other famous routing based attack is Black-hole attack [41] [20] where attackers try to send packets to benign nodes to attract benign users to send requests to the attacker. Then the attacker will drop or suspend the packets.

3.6.3 False Message Injection

In false message injection, attacker will send fake information to other vehicles like urgent brake BSM packets. If benign node trusts the attacker, it will immediately brake that may cause a lot of damage. It can not only cause property damage, but also lead to traffic jams. If it happens in highways, the damage can even get larger cause the behind car may collide your car with very high Velocity.

Chapter 4: System Model

In my proposed model, I provide a decentralized trust scheme to calculate Trust in the V2X system. There is no central server in my proposed model. I adopted a recommendation-based scheme in my system. Each vehicle in the vehicular network is represented as a node in the network. The overview of my framework is shown in Figure 4.2. I have the following procedure for a car to get indirect trust scores. Figure 4.1 shows an example of a V2X network. In the network, some cars are connected to each other. A car can send a BSM packet to another car only if they are directly connected to each other. For example, in Figure 4.1, vehicle A can send packets to vehicles B, C, D and E. They can share information with each other and alert messages such as emergency braking packet. If an accident occurs in front of vehicle B, it can send an alert message to the vehicles behind it, prompting them to brake so they do not collide with vehicle B. However, malicious vehicles can cause problem by sending inaccurate packets or any malicious content to benign cars. Thus, we have to figure out a way to trust vehicles. A packet coming from a well-trusted source is more reliable than one from an unknown source. As a result, I want to find a way for cars to trust each other. In Figure 4.1, Suppose vehicle A wants to obtain a trust score for vehicle G. There is no direct connection. How can vehicle A achieve this?

1. Request a recommendation from its neighbor.
2. The neighbor receiving the request packet will check if the target node is in its local trust table.

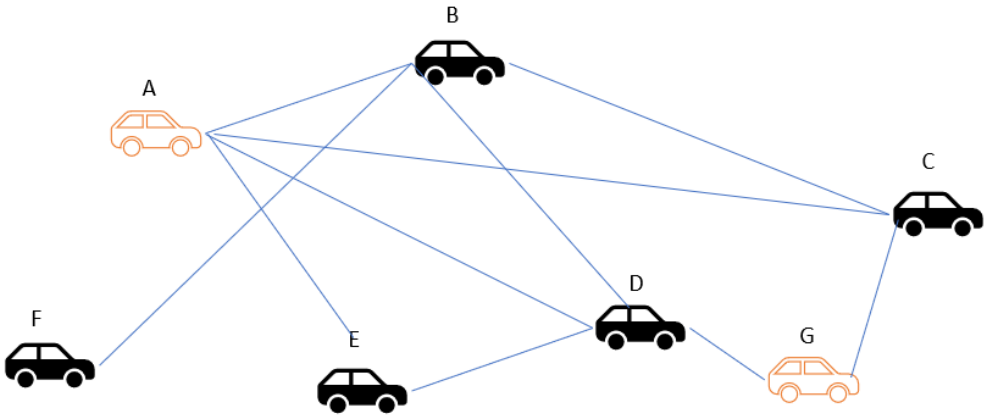


Figure 4.1: V2X Example

(a) If the target node is in the neighbor's local trust table, then return the path and trust score to the source node.

(b) If the target is not in the neighbor's local trust table, then forward it to neighbors until the target node is found.

3. After obtaining the trust path, the source node will calculate the trust score from the path and determine the indirect score.

4. Finally, it will judge to determine if the node is benign or malicious.

4.1 Local Trust Table

Each node in the network represents an entity. The trust exists between nodes. After each node interacts with others, they will form some level of trust towards each other. In my scenario, trust is represented as a continuous number ranging from 0 to 1. A trust score of "1" indicates full trust, while a score of 0 means complete distrust. Trust is a relative value rather than an absolute one. Since 0.5 is the middle of the trust score. If the score is below 0.5, it suggests that the node may view the other node as

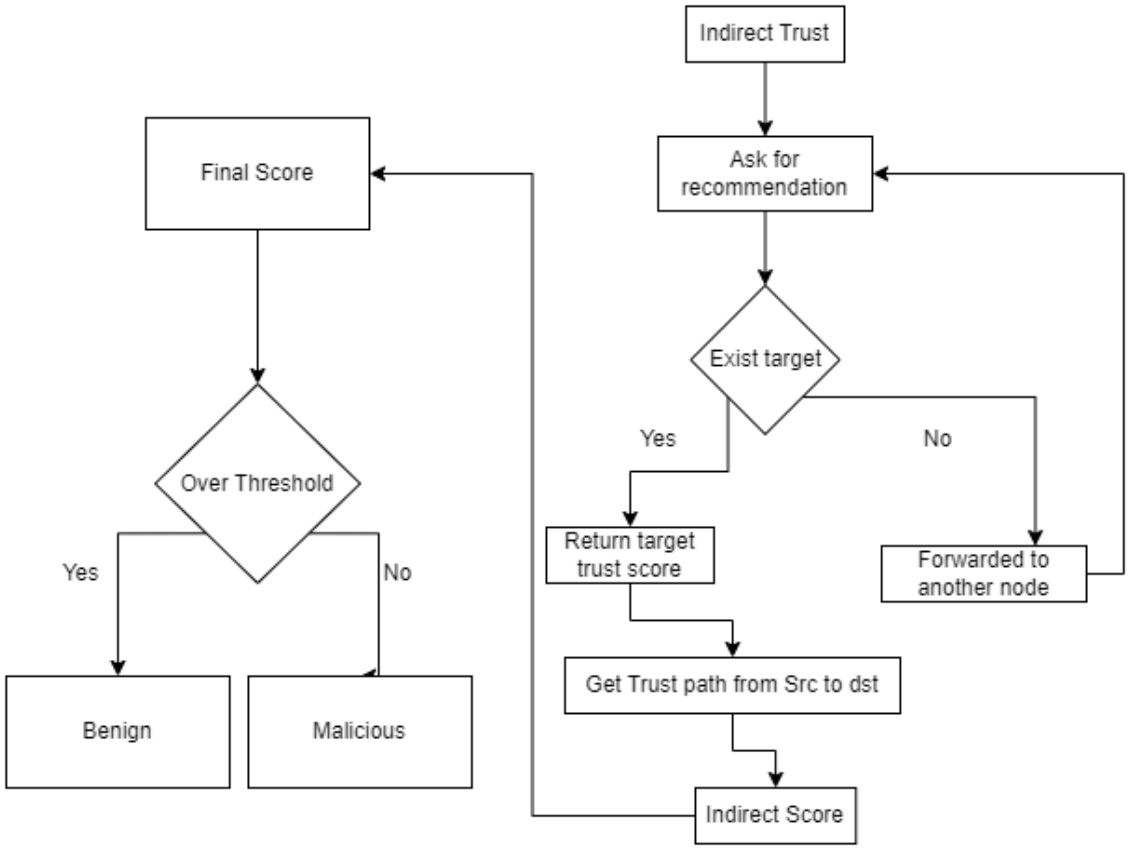


Figure 4.2: System

malicious, whereas a score above 0.5 indicates a benign node.

I make the assumption that each node can differentiate between benign and malicious nodes after directly interacting with them. For example, if node A has interacted with node B, A would generate a trust score for B. Assuming this score is 0.8, I understand that A trusts B to some extent but not completely. Each node in the network must have interacted with some other nodes to form these trust scores.

I store these trust scores in a "local trust table" within each node. This table operates on a queue-based system, following a first-come-first-serve approach. When the table reaches its maximum capacity and new records need to be added, the oldest record is automatically removed. In my model, I set the maximum capacity to 50 records. If there are too many records in

the trust table, it may occupy too much memory. This scheme is inspired by computer cache concepts, allowing each node to manage its table size efficiently.

Table 4.1 shows node A's personal view in its local trust table. Figure 4.3 shows the node A's connecting nodes. Node A has previously interacted with [B, C, D, E], and the corresponding trust scores were [0.7, 0.8, 0.2, 0.4] respectively.

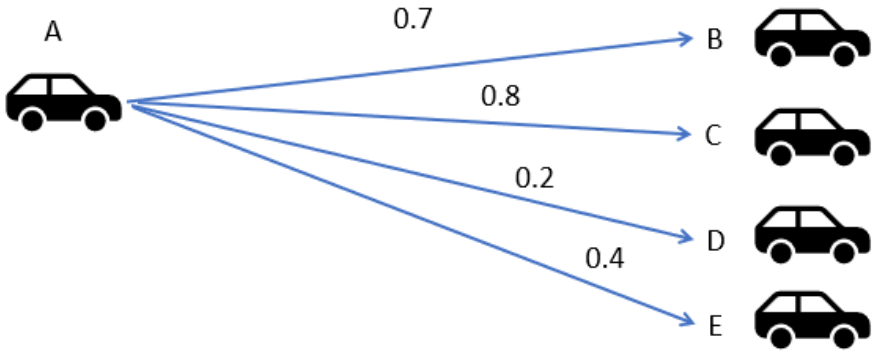


Figure 4.3: Example A

Table 4.1: Node A's local trust table

Node	Trust score
B	0.7
C	0.8
D	0.2
E	0.4

4.2 Transitive Trust

It is a common assumption that trust is transitive; however, this is not entirely true. Trust must satisfy certain conditions to be considered transitive and I refer to this as "indirect trust".

Firstly, trust is unidirectional. For instance, if A trusts B, it does not imply that B trusts A. All trust scores in my model are unidirectional. When A includes B's trust score in its local trust table, A establishes a relative trust score for B. For example, if A's local trust table shows a score of 0.6 for B, it indicates that A trusts B with a score of 0.6. I cannot derive any information about A's trust score from B's perspective in A's local trust table.

Trust can only be transitive if the recommender is trusted by the source node, and the recommender provides effective recommendations. For example, let's say A wants to repair a car but doesn't have the skills. A asks B for help because B is A's best friend. Although B also can't fix the car, B knows someone, C, who can. So, B recommends C to A for car repair. This is a good example of trust transitivity.

However, if C is not skilled in fixing cars but is instead skilled in fixing computers, then B cannot recommend C to A because it would be an invalid trust transitivity[12] [31] [1].

4.2.1 Trust Decay

In my model, I firmly believe that trust cannot be 100% transitive to other nodes [12] [36]. Therefore, I have implemented a "trust decay" function to determine the decay coefficient. By incorporating path length into the parameters, I can distinguish between short path lengths and long path lengths. As cited in prior work [36], trust decay is non-linear with increasing hops. Trust decays slowly with fewer hops, but as the number of hops increases, trust decays drastically.

I apply a real-world concept to trust, where if your friend B recommends person C to you, you may not trust C 100%, but you would still assign a

high trust score due to your friend's recommendation. However, if C then recommends a new person, D, to you, you might feel a slight distrust toward D. If D further recommends another new person, E, to you, your distrust of E would increase. As the path length increases, so does the level of distrust.

In my model, I use the following hyperbolic formula to represent the trust decay coefficient [36].

Formula 1.

$$d = 1 - \frac{1}{1 + e^{-k(l-c)}}$$

In this function, d represents the decay coefficient, while k and c are constants. L denotes path length, and k and c are used to address different scenarios. The variable k primarily determines the convergence of trust decay when l equals c. The variable c determines how many path lengths are considered in the scenario.

Since path length =c results in a value of 0.5, indicating that even benign nodes with a trust score of 1 will be multiplied by 0.5, potentially leading to a classification as a malicious node. In my model, I set k=1 and c=6. According to the small world characteristic [42], most paths can be found within 6 hops. Therefore, I assume a maximum path length of 6.

I set k equal to 1 because I believe trust decays slowly for paths of 4-5 hops and decays quickly for paths of 6 hops. The trust decay coefficient begins to decrease significantly starting from a path length of 3.

4.3 Graph-based Algorithm

To propagate trust through multiple nodes, I've reworked my approach into a graph-based algorithm. Since V2X vehicles are interconnected, we can represent them as nodes in a graph, with connections between vehicles

forming the edges. By transforming the V2X problem into a graph problem, we can apply various graph algorithms such as Longest Path Problem [66] and path-finding problem [23]. First of all, I find all paths from source node to the target node in the network. Then I calculate the path by the following formula. T(a,c) means trust path from node a to node c. For instance, as shows in Figure 4.4, if node A wants to get indirect trust score with C, A can start to find a neighbor and routing to find a path to C. If there is at least one path from A to C, A can go through this path and calculate indirect trust by formula 2. Assuming A has interacted with B and has trust score of B (0.8), B has interacted with C and has trust score of C (0.9) , then trust decay is 0.5 . In this scenario, A can get the indirect trust of C by B's recommendation. The indirect trust score will be 0.8 $\times$ 0.9 $\times$ 0.5=0.36.

As the number of cars increases, some may not connect to any others initially. In this case, *Trustcito* cannot function because it cannot obtain recommendations. Consequently, it cannot use a recommendation system to calculate the trust score. *Trustcito* have to find another solution to start with, such as a direct trust scheme, to address the problem.

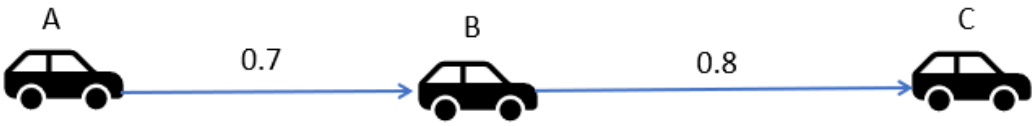


Figure 4.4: Example of A get C

Formula 2.

$$T_{a,c} = T_{a,b} \cdot T_{b,c} \cdot d$$

4.4 Multiple Paths

As more vehicles connect to each other, the potential for multiple paths between a source and destination increases. However, as I need to choose one path to be the trusted path and calculate indirect trust We need to select that path. There are many schemes for such a selection.

Zhao et al. (2013) [69] use the maximum value to calculate indirect trust while facing multiple paths, referring to the chosen path as the Most Trustable Path (MTP). In our model, I adopt this approach. I believe that individuals may place varying levels of trust in their friends, with recommendations from closer friends being more convincing than those from more distant acquaintances. However, by selecting the maximum value, I am vulnerable to Good-mouth attacks[68] which attacker can send fake recommendation with high trust score to their malicious peers. To address this issue, I dynamically monitor the path scores. If a path's score falls below a certain threshold, I discard it and select an alternative path for calculating indirect trust. This method allows us to mitigate the influence of potentially malicious recommendations on the chosen path.

Aifarez et al. (1998) [1] use the average to calculate trust over multiple paths. This method is less susceptible to good-mouth or bad-mouth attacks unless there is a high density of attackers in the environment. By computing the average value, this approach tends to reflect the collective recommendation more closely. However, it does not account for varying levels of trust among friends. I believe that in the real world, recommendations from closer friends hold greater weight than those from more distant acquaintances, even when averaged with others' opinions.

Shabut et al. (2018) [56] use the minimum value to calculate trust

while facing multiple path. This method is effective in defending against good-mouth attacks but is vulnerable to bad-mouth attacks which use fake recommendation with low trust score to benign vehicles [60]. With the minimum value, individuals can assess the minimum level of trust they can place in a node. If people adopt this method, they may be less easily deceived by malicious nodes in the environment, as it trusts fewer nodes compared to the maximum value method. However, I believe that individuals will be reluctant to trust nodes with low scores. If the vehicle's trust score is low, it means that this vehicles is not trusted by others. Thus, its recommendation cannot be a great recommendation.

Seaton et al. [54] propose an alternative method for selecting paths when faced with multiple options. Their model calculates risk, that can be reversed to represent trust. Initially, they determine the set of possible paths. Then, they calculate risk using their unique property, "extension". In their model, the "extension" property ensures that a path maintains a high risk (low trust in our context) as nodes are added to it. For example, if the risk score for the path [A–>B–>C–>G] is 0.9, and there exists an alternative path [A–>B–>C–>D–>F–>G] with a risk score of 0.8, they still choose the path with the higher risk score of 0.9. In their model, they can effectively defend against attackers from nodes D and F, which might provide inaccurate recommendations, thereby leading to inaccurate risk assessments.

4.5 Time To Live

In my model, I have added the Time To Live (TTL) feature. If a node sends a request to another node and does not receive any response within the given time, the node will search for an alternative path to seek recommendations.

This helps us defend against routing-based attacks such as black-hole attacks [41] [20].

Furthermore, I aim to enhance the redundancy and fault tolerance of my model. Since there are various reasons why a node may not receive responses from its neighbors (e.g., neighbors being exploited, network unreachability, etc.), being able to find a new path for recommendations prevents the node from getting stuck in a loop of sending requests and waiting for responses. This improvement can significantly enhance the overall performance of the system.

Chapter 5: Fault Tolerance

5.1 Fault VS Failure

In any kind of end to end device including P2P system, Vehicle to Everything(V2X) systems etc., it is imperative to distinguish fault and failures, as these terms play an important role in assessing the system's reliability and performance.

5.1.1 Faults

According to ISO document 10303-226 [46], fault is defined as a defect or abnormal condition within a system that can lead to a failure. In V2X, for example, a fault may occur due to malfunction in a single sensor that is responsible for providing accurate environmental data to an autonomous vehicle. In this scenario, the sensor fault is the potential issue, resulting a deviation from its expected behavior.

5.1.2 Failures

Failure is a combination of different defects that finally lead to software failures and results in the loss of information in critical modules , thereby making the system unresponsive. System failures can happen due to various factors, including hardware malfunctions, software bugs, network issues, or environmental challenges. Depending on the severity of the failure, the result can range from inconveniences to critical disruptions.

Felix et. al. [24], provide a that fault is a system state that can cause error. An error can finally turn into a failure which means the system

deviated from its expected behavior. In V2X, there are also some factors that will cause failure; for example, network crash. It will disrupt the continuous traffic which may cause severe damage. For instance, cars cannot transmit messages to each other. Some important messages cannot be delivered. More detail will be presented in the next section.

There are various failure types in V2X. I provide a failure classification for developers to easily detect, mitigate and tolerate the fault [53][10] .

1. Crash Failures: A component or node in the system abruptly stops functioning, resulting in a complete failure. It can be caused by many factors including software bugs, hardware failure, and network problems. Here are some common crash failures.

   (a) Memory errors: If a vehicle's onboard systems experience memory exhaustion or corruption, it could lead to critical failures, potentially resulting in safety hazards or loss of functionality.

   (b) Resource contention: Competing for system resources like CPU and network bandwidth could lead to delays in critical communications or even system crashes, impacting the ability of vehicles to exchange important safety information.

   (c) Software bugs: Flaws in the software controlling various vehicle systems could cause unpredictable behavior or system crashes, affecting the reliability of vehicle-to-vehicle (V2V) and vehicle-to-infrastructure (V2I) communications.

   (d) Hardware failures: Malfunctions in hardware components could disrupt crucial vehicle systems, compromising safety-critical functions such as braking or steering.

   (e) Network connectivity issues: Problems with the network, including

latency, packet loss, or signal interference, could lead to communication failures between vehicles and infrastructure, impacting the effectiveness of V2X communication.

(f) External Dependencies: Dependencies on external services or components, such as GPS or traffic management systems, could introduce vulnerabilities or points of failure into the V2X ecosystem.

(g) Security Vulnerabilities: Exploitable security flaws could be leveraged by malicious actors to disrupt or compromise V2X communications, potentially leading to accidents or unauthorized access to vehicle systems.

2. Omission Failures: Omission failures in a V2X could occur if critical safety features or communication protocols are not properly implemented or are inadvertently left out of the design. This could result in gaps in the V2X communication ecosystem, reducing its effectiveness in preventing accidents or coordinating traffic flow.

3. Timing Failures: Timing failures could occur if V2X components fail to meet strict timing constraints for transmitting or receiving critical safety messages. Delays or inconsistencies in message delivery could compromise the reliability of V2X communications and hinder timely responses to potential hazards on the road.

4. Performance Failures: Performance failures in a V2X could cause as bad network throughput or increased latency, impacting the responsiveness of V2X communications and diminishing the effectiveness of safety-critical applications such as collision avoidance systems or cooperative intersection management.

5. Byzantine Failures: Byzantine failures occur when components behave arbitrarily, potentially providing incorrect or malicious information to other components in the system. In a V2X, Byzantine failures could lead to miscommunication between vehicles or between vehicles and infrastructure, undermining the trustworthiness of the entire V2X network and compromising road safety[18].

5.2 Fault Tolerance

To consider fault tolerance, I may need two major classes of system properties describing system behavior: safety and liveness[25]. Safety property is to make sure that the system always runs in a "legal" state. A simple safety property can be shown as follows: Considering a V2X example, the communication must never be start when users pass the authentication. People will not be safe if communication can start without any authentication. In normal status, communication should always remain closed. Once the authentication passed, the communication can be open. An easy program can be used to explain this.

begin

*normal

If authentication=1 -> communication open

else -> communication closed

*fault

If authentication=0 -> communication open

end

On the other hand, liveness property, is to make sure that state will finally reverse. The state will finally turn into a "good" condition. Taking

the communication example again, if user pass authentication, communication can be opened. In normal status, there always exists a way to pass authentication so that can open the communication.

No matter how well the system is designed, there will be faults in the system. The goal of fault tolerance is to provide a solution when system faces some faults in order to make sure that system will not fail;i.e., the fault will not lead to failure in system. In formal fault tolerance, I need to know what fault class I need to tolerate. In V2X, I may meet different kind of faults such as crash, network errors, etc. To address these faults, I may consider solutions for each fault. However, there still may be some general solution for some faults such as redundancy.

The term redundancy means I need to have a backup plan [51]. When a system fault occurs, I have a back up plan to execute. There are 2 classes of redundancy. One is redundancy in space and the other is redundancy in time[25]. The definition of redundant in space is "If there is no fault, all executions in the redundancy part will never be reached". On the other hand, The definition of redundant in time is "If there is no fault, all executions in redundancy part will never be executed"

In my model, when I transmit trust between nodes , some faults will occur. In my model, the most common fault occur is network connectivity issues. To address network problem in my model, I adopt redundancy in my model. I choose redundancy in time. First I setup a time variable. If the program reached a timeout, means there must be a problem in the path to next node. It could not only be network problem but also due to black-hole attack. As a result, I setup a function that delete the corresponding node in trust table so that the path will be drop out. Then I will choose another path to calculate trust. I may choose the remaining path from the routing

result and select the maximum value along that path as the trust score to derive an indirect measure of trust. In normal status, I do not execute that function to make sure redundancy in time.

Chapter 6: Experiments and Analysis

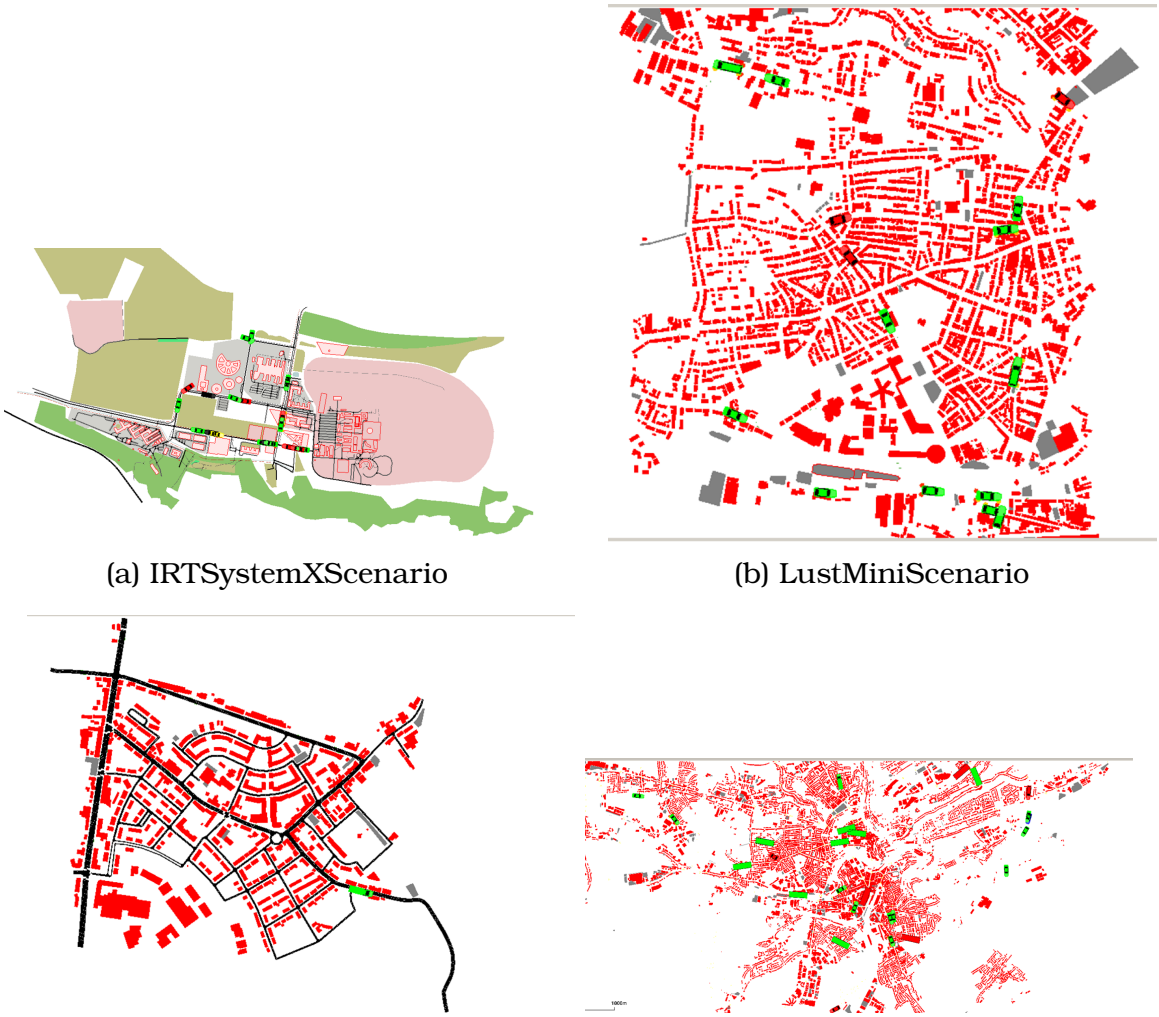
I evaluate my model using simulations. The simulator parameters are listed in Table 6.2. I used veins-F2MD[59] runs in Ubuntu 18.04 to do the experiments. Veins-F2MD is a powerful framework that usually use into simulate v2x environment. It integrates with OMNeT++ and SUMO, providing a robust simulation platform. F2MD is the framework for misbehavior detection. I choose the following maps in the veins-F2MD. I have 3 different experiments. One is a comparison of different map . A second one modifies the number of vehicles and the final one is modifies the attacker percentage.

1. IRTSystemXSCenario: The original map is in Veins-F2MD, depicting the IRTSX campus. Figure 6.1a

2. LustMiniScenario: The smallest Luxembourg map depicts a partial area of Luxembourg, inheriting some traffic data from the LustScenario. However, its scale is smaller compared to it. (See Figure 6.1c.)

3. LustNanoScenario: The smaller Luxembourg map retains only a few parts of the larger Luxembourg map, with minimal traffic depicted on it. Figure 6.1b

4. LustScenario: The entire Luxembourg map, also utilized in Veins-F2MD, is quite extensive, featuring a complex traffic layout. (See Figure 6.1d.)



(a) IRTSystemXScenario

(b) LustMiniScenario

(c) LustNanoScenario

(d) LustScenario

Figure 6.1: Different maps

For comparing against different vehicle percentage, I setup different initial vehicle numbers in simulator. Then I compare nodes and edges generated by the model as well as the query-hit rate (Detailed in Sec 6.2). The vehicle number I used is 50 vehicles versus the full route vehicles that is the default setup in the simulation. Each map's vehicle number is shown in Table 6.5

To compare different attack percentages, I choose various malicious rates to simulate using LustScenario [13]. Specifically, I select 10% as a very low

attacker percentage, 30% for a higher attacker percentage, 60% for more than half of the nodes being attackers, and 90% for nearly all nodes being attackers. I then compare the outcomes and performance in these four scenarios.

Table 6.1: Attacker percentage

Attacker percentage	Explain
0.1	Very low attacker percentage
0.3	Some attacker
0.6	More than half
0.9	Almost attacker

Table 6.2: Experiment Parameters

Simulation Time	3600s
Malicious Rate	10% / 30% / 60% / 90%
Initialy trust for benign	Random 0.51-1
Initialy trust for Malicious	Random 0-0.49
Judge Threshold	0.3
Query times	1000
k	1
c	6
Attack Type	Mix

I will do the following analysis in experiments:

1. Precision, Recall, and F1 (Detailed in Sec 6.3)

2. Query hit rate (Detailed in Sec 6.2)

6.1 Generate the Trust

In order to evaluate my model, I first used a script to randomly assign different direct trust scores to each node in the initial stage. I randomly assign trust score higher than 0.5 to benign nodes and lower than 0.5 to

malicious. This is to fit my assumption that each node can distinguish the benign node and malicious node after direct interactions.

Then I start my simulator. I randomly select one pair of source and destination vehicle to calculate trust . The source node will ask for recommendation from neighbors and get the trust score.

I set up the threshold to determine the node's label. If the final trust score obtained from the path is larger than the threshold, I label it as a benign node; otherwise, I label it as malicious. In my model, I set the threshold to 0.3. This decision is influenced by my trust decay coefficient, which limits the trust score a malicious node can achieve since their initial trust score is below 0.5. If path length is 6, it will get 0.5 decay from the decay function. As a result, attacker cannot gain higher than 0.3 in any case. I repeated this process 1000 times to calculate accuracy.

6.2 Query hit rate

In each query in my model, if the node gets the final trust score from recommendation, I mark it as successful query. Then I count every successful query and divide the total query to get the rate. In a highly connected map, as there are numerous edges, the query hit rate will be high. However, in less connected map, with fewer edges, the query hit rate will be lower.

6.3 Results

The Figure 6.2a, Figure 6.2b, Figure 6.2c, and Figure 6.2d represent the results for each map. The horizontal axis represents three classes: precision, recall, and F1 score, with values ranging from 0 to 1. I use box plots to show the deviation in each simulation. A larger box indicates greater

deviation.

I set up the same number of vehicles (50 vehicles) for each map. Since different maps have different characteristics, the results vary accordingly. Table 6.4 shows the different nodes and edges generated by the simulation with 50 vehicles. More edges generally result in a higher query hit rate.

Additionally, Table 6.5 displays the different nodes and edges in the simulation with full vehicles.

Table 6.3: True positive matrix

	Benign (Model)	Attacker (Model)
Benign (Dataset)	True positive	False negative
Attacker (Dataset)	Fasle Positive	True Negative

Table 6.4: Nodes and Edges in Each Map with 50 vehicles

	Nodes	Edges
IRTSystemXScenario	22	238
LustMiniScenario	44	139
LustScenario	44	167
LustNanoScenario	24	34

I employed Precision, Recall, and F1 score as evaluation metrics for my model. True positives are instances where a node labeled as benign is indeed benign, as detailed in Table 6.3 Through my research and experimental findings, *Trustcito* has conclusively showcased the efficacy of indirect trust mechanisms in discerning between malicious and benign vehicles within V2X networks.

$$Precision = \frac{TruePositive}{TruePositive + FalsePositive}$$

$$Recall = \frac{TruePositive}{TruePositive + FalseNegative}$$

$$F1 = 2 * \frac{Precision * Recall}{Precision + Recall}$$

6.3.1 Experiment1-Different Maps

In the IRTSystemXScenario, I achieved approximately 80% precision and nearly 100% recall. The F1 score is higher than 0.8 which is the highest F1 score in all maps. This is because there are most edges and fewest nodes in IRTSystemXScenario.

In the LustNanoScenario, precision exceeded 90%, while recall rates ranged from 75% to 92%. In this map, there are the fewest edges then it exhibits the highest deviation. This observation leads me to conclude that the fewer the edges, the greater the deviation. The F1 score is range from 80% to 95%.

In the LustMiniScenario, precision also exceeded 90%, but recall rates were below 50%, resulting in an average F1 score of around 0.6.

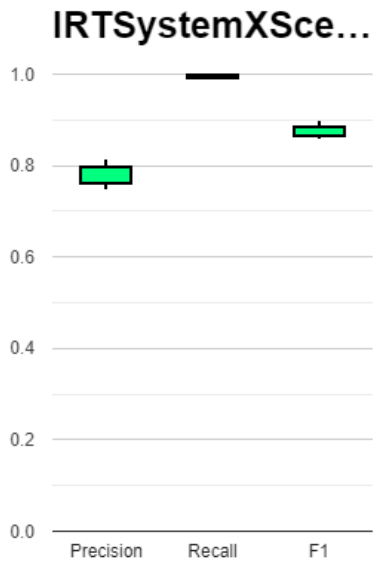
Lastly, in the LustScenario, precision exceeded 90%, with recall rates below 50%, The F1 score is higher than 60%.

Based on the results, it's evident that my model can function effectively across various maps. The generation of nodes and edges by *Trustcito* varies depending on the map used, consequently yielding diverse outcomes.

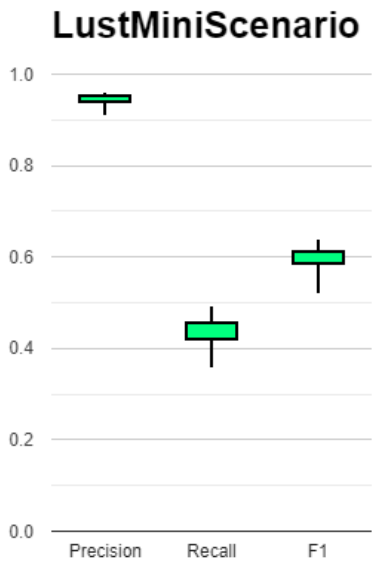
In analyzing the Query-Hit rate across different scenarios Figure 6.3, it's evident that the characteristics of each map and the density of vehicles within them play crucial roles in determining the effectiveness of information dissemination.

In the case of the IRTSystemXScenario, where the query-hit rate is the highest, the sparse distribution of nodes coupled with a dense network of edges facilitates efficient communication among vehicles. This scenario likely represents an urban environment with well-established infrastructure and a high density of vehicles, allowing for reliable information exchange.

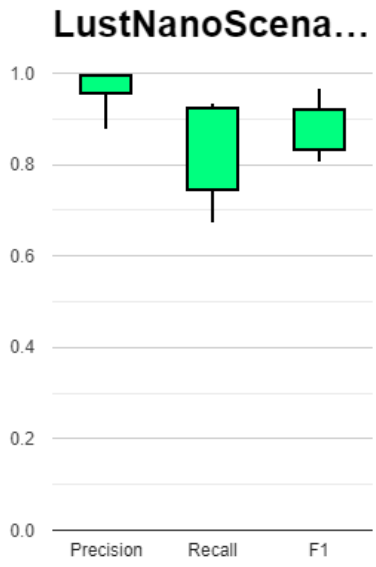
Conversely, the LustNanoScenario, with its lowest query-hit rate, reflects



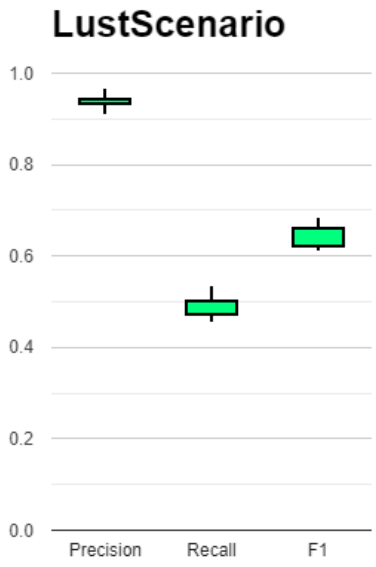
(a) IRTSystemXScenario



(b) LustMiniScenario



(c) LustNanoScenario



(d) LustScenario

Figure 6.2: Different map with 50 vehicles simulated

a scenario with limited connectivity and sparse vehicle presence. In such environments, vehicles may struggle to establish reliable communication links, leading to a reduced ability to exchange information effectively.

For LustScenario and LustMiniScenario, due to the same nodes and similar edges, these two maps exhibit similar query-hit rates.

The impact of different maps cause different nodes and edges generate by *Trustcito* leading to different query-hit rate. While the IRTSystemXScenario benefits from its dense network topology, scenarios like LustNanoScenario face limitations due to their sparse connectivity.

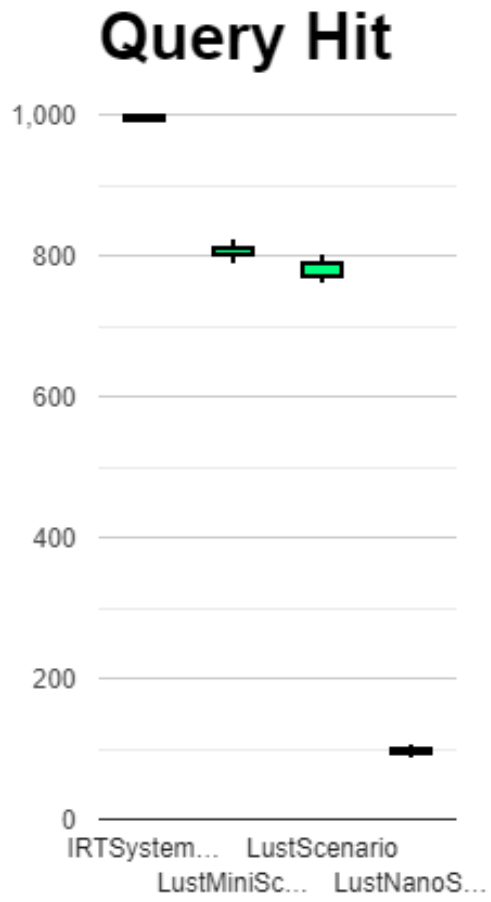


Figure 6.3: Query Hit on Different map with 50 vehicles

6.3.2 Experiment2- Different Numbers of vehicle

The other factor that affects the query-hit rate is the number of vehicles. In Table 6.5, I used the original vehicle numbers to simulate. Then I compared the results with 50 vehicles simulated. In Figure 6.4c, Figure 6.4b, and Figure 6.4a, 'partial route' means the 50-vehicle simulation, while 'full route' means the default vehicles in the simulation. Table 6.5 shows the vehicle numbers for each map. The vertical axis represents the successfully

hit rate in 1000 simulations. As the number of nodes increases, the query-hit rate drops. This is because we have set the maximum path length to 6. As more vehicles join the network, even though many cars connect to each other, if they cannot establish a connection within 6 hops, they still cannot obtain a trust score through *Trustcito*.This results in a drop to the query-hit well.

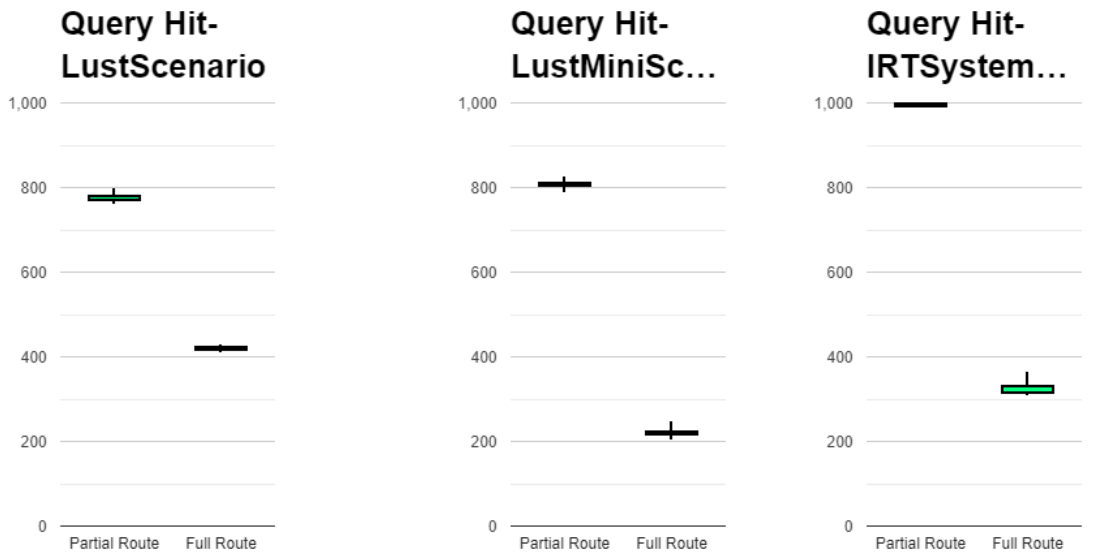
In the IRTSystemXScenario, with a simulated vehicle count of 50, the query-hit rate nearly reaches 100%. However, when the number of simulated vehicles increases to 1069, the query-hit rate drops to only 30%-40%. This outcome indicates that as the number of vehicles increases, the nodes generated by *Trustcito* also increase in number. Despite having numerous edges, the system still struggles to achieve a very high query-hit rate.

In LustMiniScenario and LustScenario, initially, both maps achieve an 80% query-hit rate. However, as the number of vehicles increases, their query-hit rates drop to 20% and 40% respectively. This outcome further highlights that as the number of nodes increases, the query-hit rate may decrease.

Comparing simulations with different vehicle counts across scenarios, it's evident that increasing the number of vehicles leads to a larger number of nodes, which in turn impacts the query-hit rate. This relationship is observed across scenarios, such as the IRTSystemXScenario (Figure 6.4c), where a higher vehicle count results in a significant drop in query-hit rate, despite having numerous edges. Similarly, in LustMiniScenario (Figure 6.4b) and LustScenario (Figure 6.4a), the query-hit rates decrease as the number of vehicles increases, indicating a correlation between node size and query-hit rate.

Table 6.5: Nodes and Edges in Each Map with all vehicles

	Nodes	Edges	vehicle numbers
IRTSystemXScenario	545	9607	1069
LustMiniScenario	161	539	184
LustScenario	317	3619	923
LustNanoScenario	24	37	55



(a) LustScenario (b) LustMiniScenario (c) IRTSystemXScenario

Figure 6.4: Query-Hit for each map

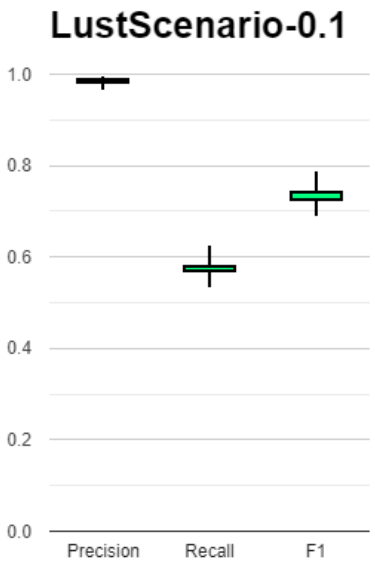
6.3.3 Experiment3-Different Malicious Rate

At various attack percentages, represented in Figure 6.5a (10%), Figure 6.5b (30%), Figure 6.5c (60%), and Figure 6.5d (90%), distinct values for precision, recall, and F1 score are observable. Specifically, at a malicious rate of 10%, the system achieves the highest precision and recall rates. However, at a malicious rate of 90%, the precision and recall values decrease significantly, indicating a decrease in the system's ability to accurately detect benign instances. It is obvious that recommendation is ineffective due to

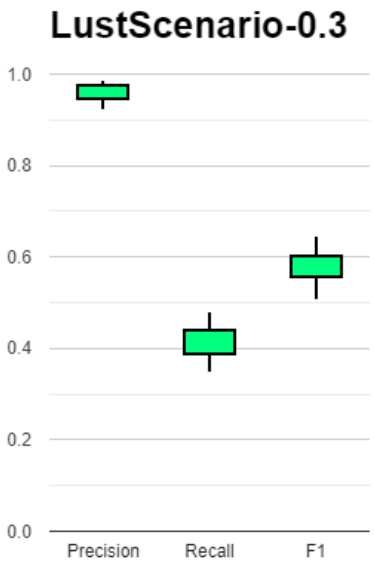
the high attacker percentage.

In scenarios where the attacker percentage is 10%, I consistently achieve over a 70% F1 score with nearly 99% precision using *Trustcito*. This demonstrates the exceptional performance of *Trustcito* when the attacker percentage is low. The effectiveness of *Trustcito* stems from its recommendation-based system, where the accuracy of recommendations plays a pivotal role. With fewer attackers present, *Trustcito* can rely on accurate recommendations to calculate precise indirect trust scores.

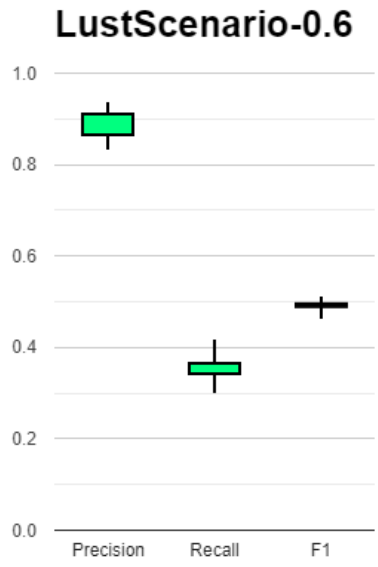
However, as the attacker percentage increases, the effectiveness of recommendations diminishes. With more attackers sending inaccurate recommendations, the reliability of the recommendation mechanism is compromised. In scenarios where the attacker percentage reaches 90%, nearly all actors in the environment are attackers. Consequently, the F1 score obtained is less than 20%, highlighting the significant impact of increasing attacker percentages on the performance of *Trustcito*.



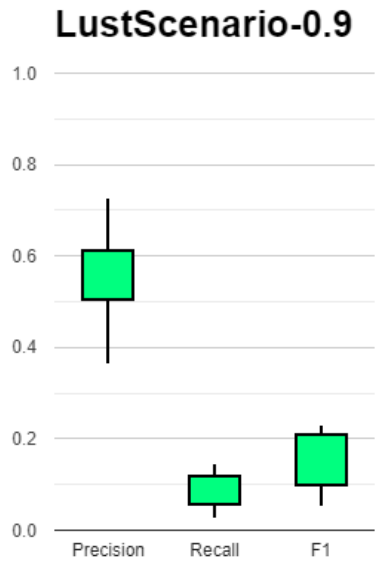
(a) LustScenario-0.1



(b) LustScenario-0.3



(c) LustScenario-0.6



(d) LustScenario-0.9

Figure 6.5: Different attack percentage with 50 vehicles simulated

Chapter 7: Conclusion

In conclusion, I found that map characteristics and the number of vehicles will affect the nodes and edges generated by *trustcito*. Due to different nodes and edges, I will get different results. More edges and fewer nodes lead to a higher query-hit rate. In terms of accuracy, I found that map characteristics and malicious rates will also affect accuracy. These findings are detailed in chapter 6.

The development and implementation of the *trustcito* framework in the V2X environment have significantly contributed to enhancing the trustworthiness and security of V2X communication networks. This provides an alternative approach to calculating trust relationships when direct trust is unavailable, thereby confirming my hypothesis that indirect trust can also be an indicator. The *trustcito* framework offers users the flexibility to assess trust in situations where traditional direct trust measurements are not feasible, thus enhancing the reliability and robustness of V2X communication systems.

However, my model has its limitations. Specifically, as the number of nodes and edges within the network grows, the computational time and resource consumption required to calculate trust scores increase substantially. This scalability issue poses a challenge for the real-world deployment and adoption of my framework. In the future, I will focus on improving the algorithm. I may use better algorithm to make the process faster and consume fewer resources. By continuing my efforts to improve my model, I can advance the safety and efficiency of future transportation systems.

Bibliography

[1] Alfarez Abdul-Rahman and Stephen Hailes. A distributed trust model. In *Proceedings of the 1997 workshop on New security paradigms*, pages 48–60, 1998.

[2] Woojin Ahn and Ronny Yongho Kim. Distributed triggered access for bsm dissemination in 802.11 bd v2v networks. *Applied Sciences*, 10(1):311, 2019.

[3] Aljawharah Alnasser and Hongjian Sun. Global roaming trust-based model for v2x communications. In *IEEE INFOCOM 2019-IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*, pages 1–6. IEEE, 2019.

[4] Aljawharah Alnasser and Hongjian Sun. Trust-based model for securing vehicular networks against rsu attacks. In *IEEE INFOCOM 2021-IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*, pages 1–6. IEEE, 2021.

[5] Aljawharah Alnasser, Hongjian Sun, and Jing Jiang. Cyber security challenges and solutions for v2x communications: A survey. *Computer Networks*, 151:52–67, 2019.

[6] Aljawharah Alnasser, Hongjian Sun, and Jing Jiang. Recommendation-based trust model for vehicle-to-everything (v2x). *IEEE Internet of Things Journal*, 7(1):440–450, 2019.

[7] Moreno Ambrosin, Lily L Yang, Xiruo Liu, Manoj R Sastry, and Ignacio J Alvarez. Design of a misbehavior detection system for objects based shared perception v2x applications. In *2019 IEEE Intelligent Transportation Systems Conference (ITSC)*, pages 1165–1172. IEEE, 2019.

[8] Mohammad Raashid Ansari, Jonathan Petit, Jean-Philippe Monteuuis, and Cong Chen. Vasp: V2x application spoofing platform.

[9] Arijita Banerjee, Sarmistha Neogy, and Chandreyee Chowdhury. Reputation based trust management system for manet. In *2012 Third International Conference on Emerging Applications of Information Technology*, pages 376–381. IEEE, 2012.

[10] Rida A Bazzi and Gil Neiger. Simplifying fault-tolerance: providing the abstraction of crash failures. *Journal of the ACM (JACM)*, 48(3):499–554, 2001.

[11] Raymond Chood. V2x security: A case study of anonymous authentication.

[12] Bruce Christianson and William S Harbison. Why isn't trust transitive? In *International workshop on security protocols*, pages 171–176. Springer, 1996.

[13] Lara Codeca, Raphaël Frank, and Thomas Engel. Luxembourg sumo traffic (lust) scenario: 24 hours of mobility for vehicular networking research. In *2015 IEEE Vehicular Networking Conference (VNC)*, pages 1–8. IEEE, 2015.

[14] Sanjay K Dhurandher, Mohammad S Obaidat, Amrit Jaiswal, Akanksha Tiwari, and Ankur Tyagi. Vehicular security through reputation and plausibility checks. *IEEE Systems Journal*, 8(2):384–394, 2013.

[15] Ahmed Didouh, Yassin El Hillali, Atika Rivenq, and Houda Labiod. Novel centralized pseudonym changing scheme for location privacy in v2x communication. *Energies*, 15(3):692, 2022.

[16] George Dimitrakopoulos and Panagiotis Demestichas. Intelligent transportation systems. *IEEE Vehicular Technology Magazine*, 5(1):77–84, 2010.

[17] John R Douceur. The sybil attack. In *International workshop on peer-to-peer systems*, pages 251–260. Springer, 2002.

[18] Assia Doudou, Benoit Garbinato, and Rachid Guerraoui. Encapsulating failure detection: From crash to byzantine failures. In *Reliable Software Technologies—Ada-Europe 2002: 7th Ada-Europe International Conference on Reliable Software Technologies Vienna, Austria, June 17–21, 2002 Proceedings 7*, pages 24–50. Springer, 2002.

[19] Hassan Farran, David Khoury, Elie Kfoury, and László Bokor. A blockchain-based v2x communication system. In *2021 44th International Conference on Telecommunications and Signal Processing (TSP)*, pages 208–213. IEEE, 2021.

[20] Elahe Fazeldehkordi, Iraj Sadegh Amiri, and Oluwatobi Ayodeji Akanbi. Chapter 2 - literature review. In Elahe Fazeldehkordi, Iraj Sadegh Amiri, and Oluwatobi Ayodeji Akanbi, editors, *A Study of Black Hole Attack Solutions*, pages 7–57. Syngress, 2016.

[21] Abdurahman Fetulhak, Abdelwahab Boualouache, Sidi Mohammed Senouci, Ines El-Korbi, Bouziane Brik, and Anlay Fante. Integrating blockchain technology with pki for secure and interoperable communication in 5g and beyond vehicular networks. In *IEEE Consumer Communications & Networking Conference*, 2024.

[22] Kai Gao, Xiangyu Cheng, Hao Huang, Xunhao Li, Tingyu Yuan, and Ronghua Du. False data injection attack detection in a platoon of cacc in rsu. In *2022 IEEE International Conference on Trust, Security and Privacy in Computing and Communications (TrustCom)*, pages 1324–1329. IEEE, 2022.

[23] JJ Garcia-Luna-Aceves and Shree Murthy. A path-finding algorithm for loop-free routing. *IEEE/ACM transactions on networking*, 5(1):148–160, 1997.

[24] Felix C. Gärtner. Fundamentals of fault-tolerant distributed computing in asynchronous environments. *ACM Comput. Surv.*, 31(1):1–26, mar 1999.

[25] Felix C Gärtner. Fundamentals of fault-tolerant distributed computing in asynchronous environments. *ACM Computing Surveys (CSUR)*, 31(1):1–26, 1999.

[26] Amrita Ghosal and Mauro Conti. Security issues and challenges in v2x: A survey. *Computer Networks*, 169:107093, 2020.

[27] Sashi Gurung, Dan Lin, Anna Squicciarini, and Elisa Bertino. Information-oriented trustworthiness evaluation in vehicular ad-hoc networks. In *Network and System Security: 7th International Conference, NSS 2013, Madrid, Spain, June 3-4, 2013. Proceedings 7*, pages 94–108. Springer, 2013.

[28] Sohan Gyawali, Yi Qian, and Rose Qingyang Hu. Machine learning and reputation based misbehavior detection in vehicular communication networks. *IEEE Transactions on Vehicular Technology*, 69(8):8871–8885, 2020.

[29] Monowar Hasan, Sibin Mohan, Takayuki Shimizu, and Hongsheng Lu. Securing vehicle-to-everything (v2x) communication platforms. *IEEE Transactions on Intelligent Vehicles*, 5(4):693–713, 2020.

[30] Jiaqi Huang, Dongfeng Fang, Yi Qian, and Rose Qingyang Hu. Recent advances and challenges in security and privacy for v2x communications. *IEEE Open Journal of Vehicular Technology*, 1:244–266, 2020.

[31] Audun Jøsang and Simon Pope. Semantic constraints for trust transitivity. In *Proceedings of the 2nd Asia-Pacific conference on Conceptual modelling-Volume 43*, pages 59–68, 2005.

[32] Chaker Abdelaziz Kerrache, Nasreddine Lagraa, Carlos T Calafate, Juan-Carlos Cano, and Pietro Manzoni. T-vnets: A novel trust architecture for vehicular networks using the standardized messaging services of etsi its. *Computer Communications*, 93:68–83, 2016.

[33] Chaker Abdelaziz Kerrache, Nasreddine Lagraa, Carlos T Calafate, and Abderrahmane Lakas. Tfdd: A trust-based framework for reliable data delivery and dos defense in vanets. *Vehicular Communications*, 9:254–267, 2017.

[34] Saurabh Kumar, Sunghyun Choi, and HyungWon Kim. Analysis of hidden terminal's effect on the performance of vehicular ad-hoc networks. *EURASIP Journal on wireless communications and networking*, 2019:1–21, 2019.

[35] Xin Li, Zhiping Jia, Peng Zhang, and Haiyang Wang. A trust-based multipath routing framework for mobile ad hoc networks. In *2010 Seventh International Conference on Fuzzy Systems and Knowledge Discovery*, volume 2, pages 773–777. IEEE, 2010.

[36] Guanfeng Liu, Yan Wang, and Mehmet Orgun. Trust transitivity in complex social networks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 25, pages 1222–1229, 2011.

[37] Nai-Wei Lo and Hsiao-Chien Tsai. A reputation system for traffic safety event on vehicular ad hoc networks. *EURASIP Journal on wireless communications and networking*, 2009:1–10, 2009.

[38] Faisal Rasheed Lone, Harsh Kumar Verma, and Krishna Pal Sharma. Recommender credibility-based trust model for secure v2x communication. In *2022 5th International Conference on Computational Intelligence and Networks (CINE)*, pages 1–6. IEEE, 2022.

[39] Rongxing Lu, Lan Zhang, Jianbing Ni, and Yuguang Fang. 5g vehicle-to-everything services: Gearing up for security and privacy. *Proceedings of the IEEE*, 108(2):373–389, 2019.

[40] Zhendong Ma, Frank Kargl, and Michael Weber. A location privacy metric for v2x communication systems. In *2009 IEEE Sarnoff Symposium*, pages 1–6. IEEE, 2009.

[41] Satria Mandala, Abdul Hanan Abdullah, Abdul Samad Ismail, Habibollah Haron, Md Asri Ngadi, and Yahaya Coulibaly. A review of blackhole attack in mobile adhoc network. In *2013 3rd International Conference on Instrumentation, Communications, Information Technology and Biomedical Engineering (ICICI-BME)*, pages 339–344. IEEE, 2013.

[42] Stanley Milgram. The small world problem. *Psychology today*, 2(1):60–67, 1967.

[43] Ed Norris. Analysis of a telnet session hijack via spoofed mac addresses and session resynchronization, 2001.

[44] Bryan Parno and Adrian Perrig. Challenges in securing vehicular networks. In *Workshop on hot topics in networks (HotNets-IV)*, pages 1–6. Maryland, USA, 2005.

[45] Nathaniel S Pearre and Hajo Ribberink. Review of research on v2x technologies, strategies, and operations. *Renewable and Sustainable Energy Reviews*, 105:61–70, 2019.

[46] Mike Pratt. Introduction to iso 10303 - the step standard for product data exchange. 2001-03-01 2001.

[47] Kashif Naseer Qureshi and Abdul Hanan Abdullah. A survey on intelligent transportation systems. *Middle-East Journal of Scientific Research*, 15(5):629–642, 2013.

[48] R Rajadurai and N Jayalakshmi. Vehicular network: Properties, structure, challenges, attacks, solutions for improving scalability andsecurity.

[49] Sitharthan Ramachandran, Baseem Khan, et al. Geometric trust-based secure localization technique for resiliency of gps outage and location error in vehicular cyber-physical systems (vcps). *Scientific Reports*, 13(1):1–14, 2023.

[50] Maxim Raya and Jean-Pierre Hubaux. The security of vehicular ad hoc networks. In *Proceedings of the 3rd ACM workshop on Security of ad hoc and sensor networks*, pages 11–21, 2005.

[51] Antonino Rullo, Edoardo Serra, and Jorge Lobo. Redundancy as a measure of fault-tolerance for the internet of things: A review. *Policy-Based Autonomic Data Governance*, pages 202–226, 2019.

[52] Christian Sanders and Yongqiang Wang. Localizing spoofing attacks on vehicular gps using vehicle-to-vehicle communications. *IEEE Transactions on Vehicular Technology*, 69(12):15656–15667, 2020.

[53] Arif Sari, Murat Akkaya, et al. Fault tolerance mechanisms in distributed systems. *International Journal of Communications, Network and System Sciences*, 8(12):471, 2015.

[54] Joshua H Seaton, Sena Hounsinou, Timothy Wood, Shouhuai Xu, Philip N Brown, and Gedare Bloom. Poster: Toward zero-trust path-aware access control. In *Proceedings of the 27th ACM on Symposium on Access Control Models and Technologies*, pages 267–269, 2022.

[55] Thejmeela Seetamonee and Girish Bekaroo. Modern detection techniques of false data injection attacks in v2x communication: A critical analysis. In *International Conference on Innovations and Interdisciplinary Solutions for Underserved Areas*, pages 66–80. Springer, 2023.

[56] Antesar M Shabut, M Shamim Kaiser, Keshav P Dahal, and Wenbing Chen. A multidimensional trust evaluation model for manets. *Journal of Network and Computer Applications*, 123:32–41, 2018.

[57] Prinkle Sharma and Hong Liu. A machine-learning-based data-centric misbehavior detection model for internet of vehicles. *IEEE Internet of Things Journal*, 8(6):4991–4999, 2020.

[58] Pranav Kumar Singh, Suraj Kumar Jha, Sunit Kumar Nandi, and Sukumar Nandi. Ml-based approach to detect ddos attack in v2i communication under sdn architecture. In *TENCON 2018-2018 IEEE region 10 conference*, pages 0144–0149. IEEE, 2018.

[59] Christoph Sommer, Reinhard German, and Falko Dressler. Bidirectionally coupled network and road traffic simulation for improved ivc analysis. *IEEE Transactions on mobile computing*, 10(1):3–15, 2010.

[60] Taisuk Suh and Youngho Cho. An enhanced trust mechanism with consensus-based false information filtering algorithm against bad-mouthing attacks and false-praise attacks in wsns. *Electronics*, 8(11):1359, 2019.

[61] Rukhsar Sultana, Jyoti Grover, and Meenakshi Tripathi. A data-centric and dynamic-range based misbehavior detection approach for vanet. In *TENCON 2022-2022 IEEE Region 10 Conference (TENCON)*, pages 1–6. IEEE, 2022.

[62] Irshad Ahmed Sumra, Halabi Bin Hasbullah, et al. Effects of attackers and attacks on availability requirement in vehicular network: a survey. In *2014 International Conference on Computer and Information Sciences (ICCOINS)*, pages 1–6. IEEE, 2014.

[63] Mingshun Sun, Ming Li, and Ryan Gerdes. A data trust framework for vanets enabling false data detection and secure vehicle tracking. In *2017 IEEE Conference on Communications and Network Security (CNS)*, pages 1–9. IEEE, 2017.

[64] Dorsaf Swessi and Hanen Idoudi. A comparative review of security threats datasets for vehicular networks. In *2021 International Conference on Innovation and Intelligence for Informatics, Computing, and Technologies (3ICT)*, pages 746–751. IEEE, 2021.

[65] Nataša Trkulja, David Starobinski, and Randall A Berry. Denial-of-service attacks on c-v2x networks. *arXiv preprint arXiv:2010.13725*, 2020.

[66] Ryuhei Uehara and Yushi Uno. Efficient algorithms for the longest path problem. In *Algorithms and Computation: 15th International Symposium,*

ISAAC 2004, Hong Kong, China, December 20-22, 2004. Proceedings 15, pages 871–883. Springer, 2005.

[67] Rens Wouter van der Heijden, Stefan Dietzel, Tim Leinmüller, and Frank Kargl. Survey on misbehavior detection in cooperative intelligent transportation systems. *IEEE Communications Surveys Tutorials*, 21(1):779–811, 2019.

[68] Fu-Guo Zhang. Preventing recommendation attack in trust-based recommender systems. *Journal of Computer Science and Technology*, 26(5):823–828, 2011.

[69] Huanyu Zhao and Xiaolin Li. Vectortrust: trust vector aggregation scheme for trust management in peer-to-peer networks. *The Journal of Supercomputing*, 64:805–829, 2013.

[70] Oliver Zheng, Jason Poon, and Konstantin Beznosov. Application-based tcp hijacking. In *Proceedings of the Second European Workshop on System Security*, pages 9–15, 2009.