Summer 2022

Secured information dissemination and misbehavior detection in VANETs

Ayan Roy

SECURED INFORMATION DISSEMINATION AND MISBEHAVIOR DETECTION IN VANETS

by

AYAN ROY

A DISSERTATION

Presented to the Graduate Faculty of the

MISSOURI UNIVERSITY OF SCIENCE AND TECHNOLOGY

In Partial Fulfillment of the Requirements for the Degree

DOCTOR OF PHILOSOPHY

in

COMPUTER SCIENCE

2022

PUBLICATION DISSERTATION OPTION

This dissertation consists of the following four articles, formatted in the style used by the Missouri University of Science and Technology.

Paper I: Pages 18 - 51, Roy, Ayan, and Sanjay Madria. "Secure and Privacy-preserving Traffic Monitoring in VANETs." was published in IEEE 17th International Conference on Mobile Ad Hoc and Sensor Systems (MASS). IEEE, 2020.

Paper II: Pages 52 - 84, Roy, Ayan, and Sanjay Madria. "Distributed Incentive-Based Secured Traffic Monitoring in VANETs." was published in 21st IEEE International Conference on Mobile Data Management (MDM). IEEE, 2020.

Paper III: Pages 85 - 116, Roy, Ayan, and Sanjay Madria. "BLAME: A Blockchain-assisted Misbehavior Detection and Event Validation in VANETS." was published in the 22nd IEEE International Conference on Mobile Data Management (MDM). IEEE, 2021.

Paper IV: Pages 117 - 143, Roy, Ayan, and Sanjay Madria. "CanSafe: An MTD based approach for providing resiliency against DoS attack within in-vehicle network" has been submitted to the 2022 IEEE Conference on Intelligent Transportation Systems (ITSC 2022).

ABSTRACT

In a connected vehicle environment, the vehicles in a region can form a distributed network (Vehicular Ad-hoc Network or VANETs) where they can share traffic-related information such as congestion or no-congestion with other vehicles within its proximity, or with a centralized entity via. the roadside units (RSUs). However, false or fabricated information injected by an attacker (or a malicious vehicle) within the network can disrupt the decision-making process of surrounding vehicles or any traffic-monitoring system. Since in VANETs the size of the distributed network constituting the vehicles can be small, it is not difficult for an attacker to propagate an attack across multiple vehicles within the network. Under such circumstances, it is difficult for any traffic monitoring organization to recognize the traffic scenario of the region of interest (ROI). Furthermore, even if we are able to establish a secured connected vehicle environment, an attacker can leverage the connectivity of individual vehicles to the outside world to detect vulnerabilities, and disrupt the normal functioning of the in-vehicle networks of individual vehicles formed by the different sensors and actuators through remote injection attacks (such as Denial of Service (DoS)). Along this direction, the core contribution of our research is directed towards secured data dissemination, detection of malicious vehicles as well as false and fabricated information within the network. as well as securing the in-vehicle networks through improvisation of the existing arbitration mechanism which otherwise leads to Denial of Service (DoS) attacks (preventing legitimate components from exchanging messages in a timely manner).

ACKNOWLEDGMENTS

First and foremost, I would like to thank my Ph.D. supervisor, Dr.Sanjay Madria, for giving me the opportunity to pursue my Ph.D. degree. Without his continuous assistance and dedicated involvement in every step of the process, this would have never been accomplished. I would like to thank you very much for your support and understanding throughout the journey.

I would also like to show gratitude to my committee, including Dr. A. Ricardo Morales, Dr. Nan Cen, Dr. Venkata Sriram Siddhardh Nadendla, and Dr. Maciej Zawodniok, for taking the interest in my work and serving in my dissertation committee.

I would also like to express my heartfelt gratitude to the Department of Computer Science at Missouri University of Science and Technology, the Intelligent System Center (ISC) at Missouri University of Science and Technology, and the National Science Foundation (NSF) for supporting my research through financial aid.

The journey to the end of the degree demanded more than academic support, and I have many people to thank for listening to and, at times, having to tolerate me over the past three years. I would like to thank my colleagues and my friends for guiding me throughout the process and being there by my side when I needed them.

Finally, and most importantly, none of this could have happened without the constant support and guidance of my family. Thank you for encouraging me during the times when I felt low, and for ensuring that I did not give up. Thank you for burning the midnight oil with me, listening to me every time I needed to be heard, and making sure that I was never alone in this journey. This dissertation will serve as a testament to the unconditional love and support that my family has bestowed upon me.

TABLE OF CONTENTS

LIST OF ILLUSTRATIONS

LIST OF TABLES

1. INTRODUCTION

In the modern transportation system, the increase in autonomy among the vehicles using sophisticated sensors and actuators, such as the LIDAR, camera, and Radar, has facilitated the vehicles to observe and gather information from their surrounding (or driving environment). Furthermore, such vehicles can form a distributed network along with the nearby vehicles, thereby forming a connected vehicle environment that is widely known as Vehicular Ad-hoc Networks (or VANETs). In the coming years, most vehicles will be equipped with onboard units (OBUs), GPS (Global Positioning System), EDR (Event Data Recorder), and sensors (radar and lidar) that are used to sense traffic congestions and status, and then automatically take appropriate actions in the vehicle and relay this information through V2V or V2I within the vehicular network [1]. The OBUs also provide computational and communication capabilities and allows the exchange of Basic Safety Message (BSM) with the nearby vehicles consisting of information such as latitude, longitude, speed, elevation, and the distance of the vehicle to a nearby object and so on. The vehicles communicate with other vehicles (termed as V2V) and with the roadside infrastructures (termed as V2I) using Dedicated-Short-Range-Communication (DSRC) devices which are based on 802.11p communication technology operating at 5.9 GHz band with a bandwidth of around 75 MHz and a transmission range of around 300 meters. Many researchers have explored the feasibility of using 5g, Bluetooth and Wi-Fi as an alternative to DSRC for connected vehicle communication which is outside the scope of our dissertation. There are several scenarios where the connectivity among vehicles is leveraged to improve driver comfort and safety. In the following two sections, we will briefly discuss the application scenarios followed by the existing challenges of vehicle communication in VANETs.

Furthermore, to make the modern-day vehicles intelligent by themselves for making autonomous decisions, vehicles are equipped with hundreds of electronic control units (ECUs) that are responsible for different functions of the vehicles, such as the brakes, steering, and throttle. The ECUs within an individual vehicle form a network called the in-vehicle network (IVN), within which they share different kinds of information, that may be perceived from the driving environment or injected remotely within the system, that determines the state of the vehicle or the next action of the vehicle from a current state. Some of the popular IVNs that are known for in-vehicle communications are control area networks (CAN), local interconnected networks (LIN), and FlexRay. However, CAN is widely accepted as a defacto standard for inter-vehicle communication due to its stability and cost-effectiveness [2]. CAN is a serial communication system where multiple ECUs connected to it share the same physical communication bus. As a mechanism to prevent the collision of messages, CAN specification defines an inverse relationship between the message IDs and the message priorities. In other words, at any given time if more than one ECU participates in the message arbitration, the ECU with the lowest message ID wins the arbitration and sends the message through the CAN. Meanwhile, the other ECUs switch to listening mode and wait for the next arbitration, thus providing a non-destructive way of message communication. Furthermore, the modern vehicles are connected to various other entities outside the in-vehicle environment using Bluetooth and 3G/4G networks, such as through the infotainment system [3]. Such remote communication with the outside world aims to enhance the driver's comfort and safety, as well as facilitate keeping the various software components of the vehicle up-to-date. regular over-the-air (OTA) updates.

1.1. APPLICATIONS

The applications of the connected vehicular network in modern transportation system is as follows:

1.1.1. Traffic Information System. The requirement for any emergency vehicle, such as an ambulance or a fire truck, is to reach its destination with minimum time delay. Under such circumstances, the vehicle opts for the shortest route to its destination. However, as the transportation network is dynamic and is not known in advance, the shortest route

(ROI) may not be the fastest. This is ideal in a situation when the shortest route may be congested due to accidents and an alternate route choice for the emergency vehicle may be optimal. Under such circumstances, the information obtained from the vehicles at the ROI can be obtained to determine the traffic condition of the region in advance to reduce the delay in reaching the destination.

1.1.2. Platooning. Platooning is a system where a group or a flock of vehicles drive together in which a vehicle follows a leading vehicle by sharing information (such as acceleration and steering) through BSM. Platooning increases the capacity of the roads via. an automated highway system [4], decreased chances of a collision, lesser congestion, and high energy saving and fuel economy.

1.1.3. Safety Applications. Vehicles can share safety information including collision warning, emergency braking, lane changing warning, lane merging and road-condition warning [5] with nearby vehicles to avoid any potential hazards or crashes. This also provides more reaction time to drivers, thereby minimizing any chance of a collision.

1.1.4. Vehicle Location Estimation. The effect of environmental conditions, such as heavy fog or rain, can have a huge impact on estimating (or knowing) the location of any vehicles nearby due to unclear vision of the surrounding. Wireless information exchanged between the vehicles can allow the drivers to know the presence of a nearby vehicle and take appropriate decisions, thereby avoiding accidents.

1.1.5. Autonomous Vehicles. Over the years, one of the major objectives of installing hundreds of ECUs is to introduce autonomy within a vehicle and empower it with real-time decision-making capabilities with minimum to no human interference. The autonomous vehicles would be able to perceive their environment (such as other vehicles or obstacles) and initiate appropriate decisions (such as braking, lane changing, accelerating/decelerating) under a given condition by ensuring public safety. Furthermore, autonomous vehicles along with a connected vehicle environment would enhance the formation of an ecosystem that would minimize crashes/accidents, and congestion, and improve public comfort and safety.

1.2. CHALLENGES

In the context of a V2X communication model, one of the major area of concern is to validate the credibility or the authenticity of the information that is exchanged between the vehicles. As the action of a vehicle is heavily dependent on driving pattern of the surrounding vehicle, any false information injected within the network can lead to inaccurate decision making, which in turn can lead towards a potential collision. Such a scenario can severely impact the decision making of the vehicles in an autonomous ecosystem that involves minimum to no human intervention. Added to it, if majority of the information obtained from the vehicles are compromised or falsified, it disrupts the decision making capability of any traffic monitoring system that considers vehicle information for determining the event of the region. Furthermore, the information disseminated by the vehicle to nearby vehicles can also lead to various privacy issues, such as unauthorized tracking of a vehicle by an attacker or leaking confidential information. The threat model (i.e the security and privacy issues) that are addressed in our dissertation has been described briefly in the following subsections:

1.2.1. Data Fabrication/ False Message Injection. In this attack, the attacker can inject false or fabricated information with the VANET. For example, it may record certain portion of the road as congested when it is non-congested and vice-versa for personal interest. As a result, such information can be taken as an input to the traffic monitoring system by the infrastructure and provide incorrect route as an optimal route. This can increase the travel time or travelling distance of a vehicle. Additionally, the attacker (or malicious vehicle) can also report false GPS location, velocity information and other vehicle parameters to the authority in order to disrupt the decision making process of the traffic monitoring system.

1.2.2. Masquerading. The purpose of an attacker is to avoid getting detected after an attack. Masquerading attack (or impersonation attack) is an active attack where the attacker tries to impersonate as other vehicle while performing the attack. For example, the attacker injects false information within the network using a different identity (while concealing its own identity) in order to avoid getting detected. It is one of the easiest attack [6] that can have a high impact within the network.

1.2.3. Sybil and Collusion Attack. In a sybil attack, an attacker creates multiple fake identities of a vehicle and reports manipulated information within the network. Such an attack can highly impact the decision making process of any nearby vehicle or any traffic monitoring system. For example, a malicious vehicle performing a sybil attack can generate a fake identity and report a false location information to the traffic monitoring system. Also, a leading attacking vehicle within a platoon can broadcast fake location, acceleration or braking information to the vehicles following it that can lead to abrupt lane change, emergency braking or abrupt deceleration for the following vehicle, which may result in an accident.

The attackers can also perform the collusion attack, where multiple malicious vehicles within a region report similar fabricated event, velocity and GPS information to the traffic monitoring system, persuading it to make inaccurate traffic decisions. As the vehicle within an ROI forms a small network at a given time, the attack can have a much severe impact if majority of the vehicles within the region collude in providing inaccurate information. Such a scenario is also possible through attack propagation. For example, the vehicle closer to an event maliciously report inaccurate information while the vehicles far away receive the malicious information and spreads it to the nearby vehicles non-maliciously. Thus, the inaccurate information propagated within the network can have a significant impact within the network.

1.2.4. Denial of Service. Denial of Service is an active attack in VANET where any event that is recorded by a malicious vehicle is not shared with the traffic monitoring system or with the nearby vehicle. Such lack of information can significantly impact the decision making process of any traffic monitoring system.

1.2.5. Packet Snooping Attack. Packet snooping attack is a passive attack that leads to a breach in the privacy of the information across the network. In this attack, the attacker tries to snoop into the data packets of the nearby vehicles which may lead to leaking of confidential information of the vehicles as well as lead to unauthorized tracking of the vehicles.

In the context of in-vehicle network, one of the major area of concern is that CAN is highly susceptible to remote security attacks, as it does not have any authentication mechanism in place. Under such circumstances, it is not difficult for an attacker to connect an external device and snoop into the data packets of the CAN. Furthermore, it can reverse engineer the information collected [7] (more commonly called as reconnaissance) and inject information within the CAN. The practicability of remote injection inside CAN has been demonstrated in [8], in which the authors were able to control the braking and steering mechanism of a vehicle through remote injection of data packets. Thus it is evident that a remote attacker can always injecting a message with the lowest ID at every time interval which in turn can deny any legitimate ECUs from sending important information to other ECUs. Therefore, an attacker can exploit the strict relationship specification between the message ID and the priority in CAN and effectively perform a Denial of Service (DoS) attack within the in-vehicle network, that leads to significant issues concerning the safety of the passengers and its surrounding.

1.3. DISSERTATION SUMMARY

This dissertation is composed of three papers presented in the publication format of the conference wherein they were published addressing the aforementioned objectives in the previous sections.

Paper I entitled "Secure and Privacy-preserving Traffic Monitoring in VANETs" presents a centralized secured and privacy-preserved approach to message exchange, event validation, and malicious vehicle identification from the ROI (even if the majority of the vehicles are malicious) by leveraging existing PKI algorithms, digital signature algorithms and information obtained from the vehicles at the ROI.

Paper II entitled "Distributed Incentive-Based Secured Traffic Monitoring in VANETs" presents a distributed game-theoretic incentive-based approach to message exchange, event validation, and malicious vehicle identification from the ROI (even if a majority of the vehicles and less than the majority of the RSUs are malicious) by leveraging the concept of clusters formation among the vehicles and Byzantine Fault-tolerant Paxos algorithm.

Paper III entitled "BLAME: A Blockchain-assisted Misbehavior Detection and Event Validation in VANETS" presents an event validation and malicious vehicle identification algorithm that leverages the radar information obtained from the vehicles and calculates the total number of vehicles at the ROI from the information obtained from the vehicles, that is made transparent at the vehicular level by incorporating the concept of the blockchain (accommodating less than majority malicious RSUs).

Paper IV entitled "CanSafe: An MTD based approach for providing resiliency against DoS attack within in-vehicle networks" presents an MTD-based solution to prevent the DoS attack within the in-vehicle network by reshuffling the relationship between the message/arbitration ID generated in a separate module, referred to as the Shuffling Module, and their priorities at every instant of time to confuse the attacker about the message priorities (as opposed to the specification of the CAN bus).

1.4. FUTURE WORK

As a future scope, we would like to design an intelligent Intrusion Detection System (IDS) or anomaly detection system using machine learning algorithms that can be deployed within the Control Areal Network (CAN) bus of the vehicle. Such an IDS can be leveraged to detect any falsified data that may have been injected by an attacker or any anomalous data that it receives in the form of BSM from nearby vehicles through V2V communication, thereby incorporating many real-time applications that are of extreme importance in an autonomous vehicle.

Our goal is also to design a strategy that can withstand DoS attacks even under the influence of multiple injection attacks within in-vehicle networks.

2. LITERATURE REVIEW

2.1. CONNECTED VEHICLE (V2X) SECURITY

In general, the VANET architecture is subdivided into 3 separate planes: (i) Vehicular Plane: consisting of the vehicles that interact with other vehicles using V2V communication, generates the information from the ROI and interacts with the RSUs using V2I communication, (ii) RSU plane: consisting of the road side infrastructures or the RSUs which are deployed within the region, that interacts with the vehicles as well as with the central authority (if any) and, (iii) Central plane: mainly consisting of the trusted authorities/central authority, key distribution center etc. The scope of this dissertation is to highlight the various security and the privacy issues for information dissemination, event validation, and malicious vehicle identification in VANET. Data emerging from one vehicle is broadcast to the nearby vehicles for validation and deciding the traffic scenario of the region. In this section, we briefly elaborate the various commonly used methods which are adopted to tackle the security and privacy challenges that are listed in the threat model along with their shortcomings. The methods has been classified into 2 major categories: (i) Centralized Solutions and (ii) Distributed Solutions.

2.1.1. Centralized Solutions. Li et. al. [9] proposed a reputation based announcement scheme using a trusted centralized reputation server for VANET, where the credibility/authenticity of the message generated by a vehicle is determined by its neighboring vehicles. Every vehicle on receiving a message from nearby vehicle determines the authenticity of the certificate attached in the message that is received from the reputation server and checks for the time discounted reputation score. If valid the message is considered reliable. For the authenticity of the message itslef, a receiving vehicle checks if the message matches with its own event recorded. If it does not it reports a feedback. The accuracy of the reputation calculated in the proposed approach relies on the assumption that the proportion of malicious vehicle with respect to the number of vehicles reporting the event is small which makes it robust against ballot stuffing or bad mouthing attack. However, in modern transportation attacks can be propagated from one vehicle to the another. In a smaller network, there is a high chance of compromising majority vehicles. Also, the percentage of malicious vehicles surrounding a target vehicle may be high enough to degrade its reputation.

Similarly, a threshold based event validation approach has been proposed by Taie et. al. in [10]. In the proposed approach every vehicle is loaded with a on-board unit (OBU) monitoring camera to collect picture evidence of an event from ROI. Thereafter, it generates the traffic report consisting of the vehicle information such as the vehicle id, current time, position,, direction and so on. Furthermore, the monitoring report is transmitted to the nearby OBU and also the RSU for validation. If the monitoring event is reported by a threshold number of vehicles, it is considered authentic else it is considered a fake message. The proposed model is susceptible to ballot stuffing or bad mouthing attack if the malicious vehicles collude to report the same incorrect event. Furthermore, the malicious vehicles colluding at the ROI need not be in majority if the threshold is lesser than the majority count.

A threshold based privacy-preserving road condition monitoring system has also been proposed in [11] where the vehicles report the reports the road conditions to the cloud server in an encrypted format. The road condition information sent by the vehicles should specify the location of the road where the event was reported. Thereafter, the cloud server is allowed to compare the reported ciphertexts from the vehicles and inform the RA to take appropriate decisions if the reported number of road conditions exceeeds a pre defined number of thresholds. However, threshold based system is easier for an attacker to compromise if the threshold is not appropriate.

2.1.2. Distributed Solutions. Singh et. al [12] proposed an incentive-based trust management approach for event detection and validation among the vehicles using sharded blockchain network deployed within the RSUs, and by using smart contracts. The use of the sharded blockchain in the proposed approach reduced the propagation delay and the work-loads from the main maintained blockchain. However, the major shortcoming of the proposed framework is that, it could effectively validate the event only when the malicious nodes at the vehicle plane (vehicles) are in minority. The proposed framework is also vulnerable to collusion attack among the vehicles and the RSUs, the Sybil attack by the vehicles at the vehicle plane, and also data manipulation attack.

Yang et.al. [13] proposed a decentralized trust management scheme using blockchain in VANET. In the proposed scheme, the vehicles generate a reputation score corresponding to certain messages within the network. The messages of same kind are grouped together. Not all messages in the group are equally credible. The messages originating from the vehicles located closer to the event are considered more trustworthy that the vehicles away from the event. The aggregate credibility score of an event is calculated using the Bayesian inference. If the aggregate credibility score is higher than a pre-defined threshold, then the event is regarded as a true event and the ratings of the messages reporting the event is incremented by a factor of 1 otherwise it is decremented. In the proposed model, the RSU can receive conflicting ratings about a specfic event, on the event of which the decision is based on the majority basis. Thereafter, a joint proof-of-work (PoW) and proof-of-stake miner election method is adopted where the difficult of solving the PoW is dependent on the stake of the miner node. The major shortcoming of the proposed scheme is that a malicious vehicle reporting inaccurate event closer to the location will be considered more trustworthy than a non-malicious vehicle far away from the event. Also, as the transmission range of the vehicles cannot accommodate every possible vehicle at the ROI, it is possible that the a non-malicious vehicle may be surrounded by majority malicious vehicle (even if they form a minority in the entire network) that can drastically degrade the credibility of the messages originating from the non-malicious vehicle, thereby performing the bad-mouthing attack.

Lu et.al. [14] proposed a privacy preserving reputation management model in VANET using blockchain named BARS. In the proposed reputation management, a vehicle receives the message originating from another vehicle and determines the credibility of the message. If the receiving vehicle disputes the authenticity of the received message or discover any misbehavior (which includes detecting any fabricated information within the network), it can expose the messages to the Law Enforcement Authority (LEA) along with the evidence to make the arbitration. If the evidence if valid, the vehicle that braodcasts the forged message will be punished while the vehicle exposing the forged message is rewarded and vice-versa. However, the proposed model does not elaborate on the evidence itself that needs to be submitted from the vehicle that wants to expose another vehicle. This is important as the information within the evidence can determine the feasibility of the malicious vehicle submitting fake evidence as a misbehavior to contradict the message reported by the non-malicious vehicles. Also, the efficacy of the proposed model is dependent on the assumption that majority of the vehicles are non-malicious.

Huang et.al. [15] proposed a secured and distributed reputation management system in VANET. In the proposed scheme, the vehicles at the ROI can interact with its one-hop neighbors and allocates a reputation value to its neighbor. The evaluation of one vehicle by another vehicle is performed using the subjective logic framework. According to it, the opinion of one vehicle for another vehicle is represented in the form a tuple consisting of 4 parameters (belief, disbelief, uncertain, constant base rate) where belief is the probability that the statement is true, disbelief is the probability that the statement is false, uncertain is the individual confidence in a vehicle's knowledge on other vehicles, and the constant base rate is the existing impression without solid evidences. These metrics are leveraged to determine the reputation segment from one vehicle about its neighboring vehicle. Thereafter, the reputation sengments are sent to the Local Authority (LA) for reputation update. In the proposed scheme, if the rater vehicle has prior knowledge of the ratee vehicle, the reputation segment of the ratee is considered highly reliable. However, the proposed scheme is susceptible to ballot stuffing and bad mouthing attack.

Khelifi et al. [16] proposed a reputation-based blockchain mechanism to secure information and data forwarding as well as content caching. In the proposed blockchain framework, a look-up table is maintained to store the reputation of the authentic vehicles that are used to validate the information. However, the proposed model is vulnerable to an on-off attack where a malicious vehicle with a higher reputation injects false information or it purposefully reduces its reputation and thereafter injects correct information which is neglected using the framework. The proposed mechanism is also vulnerable to an attack led by the collusion of malicious vehicles.

DrivMan [17] is a trust management framework where the reliability of the data originating from a registered source is ensured using physical unclonable functions (PUF) and smart contracts. In the proposed framework, every vehicle registers itself with Drivman to become the constituent of the network. At the time of registration, the challenge-response pair (CRP) corresponding to its PUF is recorded by teh operator of the Drivman network. When a message from a vehicle is encountered, it verifies if the vehicle belongs to the trusted list of vehicles. If so, it invokes the PUF challenge response protocol to issue a certificate to the vehicle. Even though the proposed framework ensures the authenticity of the vehicle sending the information, it does not verify the content of the message sent. In other words, DrivMan does not deal with message validation and is not effective if a registered vehicle reports a false message to the RSUs.

A traffic event validation and trust management framework based on blockchain has been proposed by Yang et.al. in [18]. In the proposed work, a proof-of-event (PoE) based consensus mechanism is proposed to obtain a reliable information from the ROI. The traffic incident from the vehicles near the RSU is leveraged to correctly determine the traffic event of the region. Once a reported data meets the required threshold, the PoE consensus algorithm is started. Each event in the proposed framework is stored as a synopsis in a hierarchical, geographical and chronological structure to improve the efficiency of the blockchain. Also the PoE mechanism can identify th malicious behaviors within the network and prevent the spread of false traffic warnings. During the PoE strategy, the other nodes of the blockchain verify the block producer using the evidence of the traffic event description. A major shortcoming of the proposed framework is that, as it is based on threshold based validation, choosing an appropriate threshold could be difficult, as a threshold lower that the majority can lead make it simpler for the malicious vehicles to collude and disrupt the system.

A decentralized multi-agent trust management scheme has been proposed by Guleng et.al. in [19]. In the proposed scheme, each of the vehicle conducts a direct trust evaluation about its one-hop neighbors based on it behaviors and other neighbors reports. The neighbors also determine the trust evaluation of its non-one-hop neighbors considering the report of multiple one-hop neighbors of the analyzed vehicle. For the direct trust calculation a fuzzy logic based approach is adopted. According to the approach, every vehicle is provided a cooperative factor, honestness factor and a responsibility factor. The cooperative factor determines the number of messages forwarded by a given vehicle (cooperative messages), the honestness factor determines the percentages of true packets that is disseminated by a vehicle within the network whereas the responsibility factor determines the percentage of reported messages that is recorded by the trustee node itself. All these factors are combined to determine the direct trust value of the node. For the indirect trust value, Q-Learning based approach is adopted where the trust values of all nodes involved in the trust forwarding and the number of hops from the trustee nodes are considered in order to achieve an efficient and accurate trust evaluation. A major drawback of the proposed scheme is that if the percentage of malicious vehicles within the network is high enough, the accuracy of the reputation management may degrade due to ballot stuffing and bad mouthing attack.

2.2. IN-VEHICLE SECURITY

Over the past few years, a significant amount of research has been devoted to the security of the CAN in general. Most the research is based on designing a prem-emptive solution.

In Pre-emptive solutions, the main intention is to prevent any attack into the CAN of a vehicle. Machine/Deep Learning based solutions have been widely explored as potential pre-emptive solutions by many researchers. For instance, the authors in [20] have analyzed the detection performance of Pearson correlation, k-means clustering and Hidden Markov Model in an in-motion vehicle with injected speed and revolutions per minute (RPM) readings. The authors concluded that the use of fabricated (or simulated) dataset incorrectly indicates the performance of an intrusion detection technique. The authors in [21][22] [23][24][25] have proposed different supervised/unsupervised machine learning solutions for detecting intrusion as a pre-emptive measure within the CAN.

Beside the machine learning models, authors have also analyzed the feasibility of graph based solutions [26][27] for detection injection attacks. The authors in [26] have proposed a message sequence based solution to detect an abnormality in pattern between the messages exchanged between the ECUs and determine the injection attack in vehicle. However, the proposed solution relies heavily on an assumption that the normal driving behavior remains constant and any abnormal driving behavior (which may not be injected but different driving condition) is an attacked state. Furthermore, the solution does not take into account that the pattern of messages sent within the in-vehicle is a public information, and can be exploited by a sophisticated attacker. Many researchers [28][29][30] have leveraged the frequency of arrival of messages in CAN as a potential pre-emptive solution for message injection or DoS. In the proposed solutions, the authors assume that every message within the CAN arrive after periodic intervals, that is unknown to the attacker. Thus, the message injected by an attacker adds to an outlier to the periodic interval, which helps in detecting an injected message. However, it is to be noted that the behavior of the

ECUs (or the messages originating from the ECUs) is quite inconsistent with respect to their arrival time, on the advent of which, the efficacy of the proposed models reduces. Furthermore, the efficacy of the model reduces if the attacker is able to mimic the original message frequency. Researchers have also exploited the idea of information theory and entropy-based mechanism for detecting message injection within the network. Very similar to the concept of periodicity, the authors [31][32] here assume that the regular CAN messages have a stable entropy, and a deviation from normal behavior is categorized as an attack by the authors.

An IP-shuffling based MTD defense mechanism has been introduced by the authors in [33] where the authors dynamically changes the default IP address of the ECUs to introduce uncertainty or nullify any reconnaissance of an attacker. However, it is to be noted that the proposed approach serves the purpose of preserving the confidentiality of the ECU from the attacker but does not stop the attacker from performing a DoS (noted by authors also). The authors in [34] has proposed CIST to leverage the use of a fixed priority ID and a dynamic ID to nullify the reconnaissance of an attacker, and provide resiliency against replay and impersonation attack. However, the proposed mechanism does not consider DoS attack as part of their threat model, which breaks the efficacy of the system if the attacker knows the fixed priority ID and replays it at different communication cycles. The authors in [35] have constructed a deterministic finite automation (DFA) for the bit-by-bit arbitration process where a firewall deployed within the system rejects any fake ID injected by an attacker if it is not contained within the whitelist of the ECU IDs maintained inside the firewall. However, the efficacy of the proposed DFA fails if an attacker gathers information through CAN sniffing to reverse engineer and gather some ECU IDs from the whitelist as the IDs remain fixed. Such a captured ID when replayed in separate communication cycle, will be considered as an authentic state by the proposed DFA.

To summarize, it can be observed that most of the preemption mechanism can be used effectively to deny message injection within the network. However, many solutions proposed are more content-based, meaning that it relies on the content of the message to detect if the message is an injected message or not. Such solutions are considered reactive mechanism, which may serve the purpose of detecting an injection attack. However, such solutions does not guarantee resiliency against DoS attack. This is because for injecting a message, as per the CAN specification, the attacker has to win the arbitration among the contending ECUs in a communication cycle. Thus, if the attacker is able to inject the payload, it means that it has successfully won the arbitration and prevented other authenticate ECUs from sending information during that communication cycle.

PAPER

I. SECURE AND PRIVACY-PRESERVING TRAFFIC MONITORING IN VANETS

Ayan Roy and Sanjay Madria
Department of Computer Science
Missouri University of Science and Technology
Rolla, Missouri 65401
Email: ar3g3@mst.edu and madrias@mst.edu

ABSTRACT

Vehicular Ad hoc Networks (VANETs) facilitate vehicles to wirelessly communicate with neighboring vehicles as well as with roadside units (RSUs). However, an attacker can inject inaccurate information within the network that can cause various security and privacy threats, and also disrupt the normal functioning of any traffic monitoring system. Thus, we propose an edge cloud-based privacy-preserving secured decision making model that employs a heuristic based on vehicular data such as GPS location and velocity to authenticate traffic-related information from the ROI under different traffic scenarios. The effectiveness of the proposed model has been validated using VENTOS, SUMO, and Omnet++ simulators, and also, by using a simulated cloud environment. We compare our proposed model to the existing state-of-the-art models under different attack scenarios. We show that our model is effective and capable of filtering data from malicious vehicles, and provide accurate traffic information under the influence of at least one non-malicious vehicle.

1. INTRODUCTION

Vehicular Ad hoc Networks (VANETs) allow wireless communication from vehicle-to-vehicle (V2V), and vehicle-to-infrastructure (V2I) such as with road side units (RSU) for better traffic management. Using the dedicated short range communication (DSRC) protocol, every vehicle broadcasts information about traffic events such as accidents, traffic congestion, and traffic violations to nearby vehicles as well as road side infrastructures. However, the presence of malicious vehicles at the region of interest (ROI) can negatively influence the traffic monitoring of the region. Bluetooth-based traffic monitoring systems such as Clearview Intelligence's M830 leverage the unique MAC address of Bluetooth devices inside the car and their entry, exit times form the zones to determine the traffic flow. However, the malicious vehicles can perform Denial of Service attack by switching off their devices, on the event of which the device records fewer vehicles within the region. Also, a naive way of thinking to solve the problem is to collect location and velocity information from vehicles and discard inconsistent information with the RSU's location and speed. However, such a solution will not be effective where the malicious vehicles send incorrect velocity information to the RSU using V2I communication. A comprehensive review of the solutions to preserve the privacy, authenticity and security of the messages disseminated in VANET has been provided in [1]. Existing strategies such as the peer authentication model [2], threshold based or majority voting [3], and the reputation-based system [4] provide traffic monitoring by assuming the concept of adversarial parsimony [5] but it becomes challenging to validate the vehicular responses when there are majority of malicious vehicles. The majority of the vehicles can be compromised when a group of malicious vehicles forming the road network performs a collusion attack. Also, in the case of a small network, an attack can be propagated among the majority of vehicles when a compromised vehicle communicates using V2V communication to its nearby vehicles, thus, compromising them.

The remote hacking of a Jeep Cherokee in 2015 [6] highlights the feasibility of such remote attacks in vehicles. Additionally, the breach in the privacy of individual vehicle information can lead to unauthorized tracking of officials as well as vehicles identity theft. The purpose of an attacker is to profile a driver's habits based on GPS location, velocity, acceleration, and the unique ID of the vehicle. Therefore, it is necessary to ensure the anonymity of the vehicles and unlinkability of the shared information to their originating vehicles. Anonymity ensures that every vehicle remains anonymous while exchanging information whereas, unlinkability ensures the inability to trace the identity of a vehicle based on the information exchanged. Furthermore, such a system should ensure conditional privacy, meaning that the identity of the attacker can be revealed in case of a conflict.

The United States Department of Transport (USDOT) introduced Security Credential Management System (SCMS) [7] that leverages V2V and V2I communication among vehicles, and public key infrastructure (PKI) to ensure message integrity, authenticity, privacy, and interoperability. Due to the lack of misbehavior detection algorithm, if authentic vehicles misbehave and provide inaccurate traffic-related information from the ROI, there is no algorithm in place to validate the information and filter the malicious vehicles and their responses from the network, and thus, it can obtain inaccurate traffic information. This problem is elevated even more if the malicious vehicles at the ROI form the majority and try to disrupt the traffic monitoring system. To address these shortcomings, we design a global misbehavior detection algorithm and propose a secure and privacy-preserving decision making model by leveraging the PKI and an edge cloud-based infrastructure to validate traffic-related information from the ROI and filter malicious vehicles, even if they are in majority (like under collaborative or DDoS attack), within the ROI. Each edge server is associated with a different region and is connected to a centralized server. By leveraging the concept of the edge server, the decision making model is brought closer to the concerned ROI, which reduces the latency in the decision and reduces the bottleneck on the centralized server. The main contributions of the proposed work:

- Develop a privacy-preserving and secure heuristic based solution that overcomes the shortcomings of the current state-of-the-art models and validates the traffic-related information from the ROI using the recorded GPS location of each vehicle, the vehicles' velocity and encrypted neighboring vehicles' IDs under the influence of at least one non-malicious vehicle within the ROI. It is unlike the assumption of the majority of non-malicious vehicles considered in other state-of-the-art models. We also consider an event recorded by an individual vehicle because the velocity of the vehicle may not always reflect the event at that ROI (such as when a vehicle is moving with a low velocity along the service lane in a non-congested road).
- Design a dynamic data structure called the *Decision Similarity Graph* based on the vehicle location, and leverage the *Point of Conflict* concept to filter malicious vehicles within the ROI using the conflicting event recorded by any two neighboring vehicles.
- We show that the model effectively validates the traffic-related information and filters the malicious vehicles and their responses from the network. The effectiveness of the model is compared against other existing state-of-the-art models under different scenarios using the simulators VENTOS, SUMO, and Omnet++.

2. RELATED WORK

2.1. MAJORITY VOTING MODEL OR THRESHOLD BASED MODEL

In a weighted majority voting [8] model, the majority of the vehicles are considered reliable and the message generated by a vehicle closer to the event has a higher weight than the vehicle at a distance. However, this may result in an inaccurate data reporting if the vehicle closer to the event is malicious or majority of the vehicles are malicious. [9][3] have proposed threshold based data authentication schemes in which a vehicle considers a message as credible if it has been authenticated by a threshold number of vehicles. However, they are highly vulnerable to ballot stuffing, bad mouthing, collusion, sybil attack and in situation when the number of malicious or compromised vehicles authenticating an incorrect message is greater than or equal to the threshold. In privacy-preserving traffic monitoring system [10], the vehicles share their speed information perturbed with noise with the nearby vehicles, and the security of the information is ensured using homomorphic cryptosystem. The traffic scenario of the region is decided by computing the average of the speed information shared by all the vehicles in the region. However, the model inherently assumes that the vehicle does not tamper its speed information to disrupt the traffic scenario of the region.

2.2. REPUTATION BASED MODEL

[4] has proposed a reputation based announcement scheme in which the credibility of the message generated by a vehicle depends on the reputation score of the vehicle which is obtained by the reliability of its broadcast messages in the past. However, the model is susceptible to on-off attack, where a vehicle with a high prior reputation score performs maliciously or is compromised by an attacker.

2.3. PEER AUTHENTICATION MODEL

A blockchain-based reputation system [2] uses the credibility of the messages generated by a vehicle in determining the prior reputation of the vehicle, and by ratings from the nearby vehicles which has also been used in [11]. [12] proposed a peer authentication based trust management model which uses trust scores. However, it is highly vulnerable to ballot stuffing and bad-mouthing attack, in which the ratings of an individual vehicle can be influenced by nearby vehicles. The proposed model also provides incorrect traffic information when the majority of the vehicles are malicious or are compromised by an attacker.

Our proposed model differs from the above mentioned models/schemes in the following aspects: 1) involves no peer authentication, majority voting, or threshold concept, 2) can provide accurate information even when majority of the vehicles are malicious or compromised.

3. PRELIMINARIES, THREAT MODEL AND ASSUMPTIONS

3.1. PRELIMINARIES

- Edge Server: A trusted entity associated with a small region, such as a down-town in a city, which monitors accurately traffic events like accident or congestion based on the proposed heuristic, and filters malicious data and vehicles. It scales the proposed model for large VANETs using the distributed cloud concept.

- Centralized Server: Since traffic congestion in a region has a cascading effect on other surrounding regions, the decisions from different edge servers deployed in small regions are analyzed by the trusted centralized server (of a city, county or a state) to generate an overview of a traffic scenario for a large region.

- Decision Similarity Graph (DSG): The Decision Similarity Graph (DSG), represented in Figure 1, is utilized by the edge server to filter malicious vehicles within the ROI, which is explained in section IV(c). DSG is an ordered pair of the form:

$$DSG =< V, E >$$

  where V=<V_1, V_2, V_3,..., V_n> represent the n vehicles from which the responses are received and E=<E_1, E_2, E_3,..., E_n> are undirected edges that represent the neighborhood between any two vehicles.

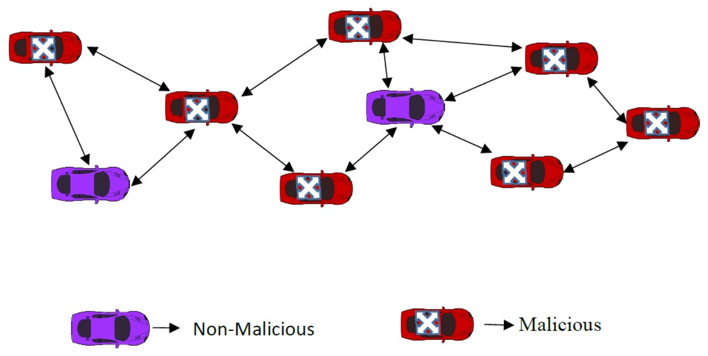


Figure 1. Decision Similarity Graph

- AES encryption algorithm [13]: Every vehicle, V_{id}, at the ROI uses the AES 128 bit symmetric encryption algorithm to generate a unique key, Key_i, which is used to encrypt the vehicular data such as ID, GPS location, and velocity before sending the data to the edge server via the RSU. Since the AES 128 bit algorithm is faster than the AES 192/256 bit key algorithms [14], and still provide enough desired security, the proposed model reduces the latency due to encryption.

- Schmidt-Samoa cryptosystem [15]: Every edge server using the Schmidt-Samoa cryptosystem generates a public key, G_{public}, and its associated private key, $G_{private}$, which is utilized by the vehicles within the ROI for a secure key, Key_i, exchange as well as for the privacy preserved V_{id} broadcast to its neighboring vehicles. Edge servers associated with different regions generate their own G_{public} and $G_{private}$ keys. We prefer [15] over RSA [16] because of its simplistic trapdoor one-way permutation. Also, unlike Rabin [17], this algorithm does not produce any ambiguity in the decryption at the cost of the encryption speed. The Schmidt-Samoa cryptosystem is preferred over the elliptic-curve cryptosystem [18],[19].

- Elgamal Digital Signature Scheme (EGDSS) [20]: Every vehicle, V_{id}, before entering into a VANET communication, is registered with the trusted centralized server using a private key, *veh_private*$_i$, and its associated public key, *veh_public*$_i$, which are generated using the EGDSS algorithm. The *veh_private*$_i$ is loaded onto the on-board-unit(OBU) of V_{id} which is used to authenticate its identity to the edge server. Instead of *EGDSS*, the digital signature algorithm (DSA) [21] could also be used without affecting the model's effectiveness.

3.2. THREAT MODEL

Malicious vehicles (1) can manipulate any recorded event (including velocity and recorded GPS location) to disrupt the decision making process, (2) may try to impersonate some other vehicle by reporting some other V_{id}, and can also send manipulated traffic-related information under different registered V_{id}s, (3) may intercept a data packet of any non-malicious vehicle, modify the information, and send it to the RSU for decision making, (4) may not send its vehicular data to the RSU to degrade the traffic monitoring system.

3.3. ASSUMPTIONS

We assume that the non-malicious vehicles always send requested information to the edge server via the trusted RSU. Every challenge and response packet is sent and received by the RSU as well as by the vehicles. If a packet originating from a vehicle is dropped due to a network problem or channel congestion, it is neglected by the proposed model. However, for the effectiveness of the proposed model, the edge server must obtain at least one non-malicious response or one conflicting neighbor of any malicious vehicle to validate the traffic-related information from the ROI if it exists. Every vehicle has a unique ID (allocated by the department of motor vehicles) of uniform length known by the edge servers. The recorded event is either congested or non-congested, defined based on the velocity of the vehicles within the ROI in the proposed model (section IV c). The proposed

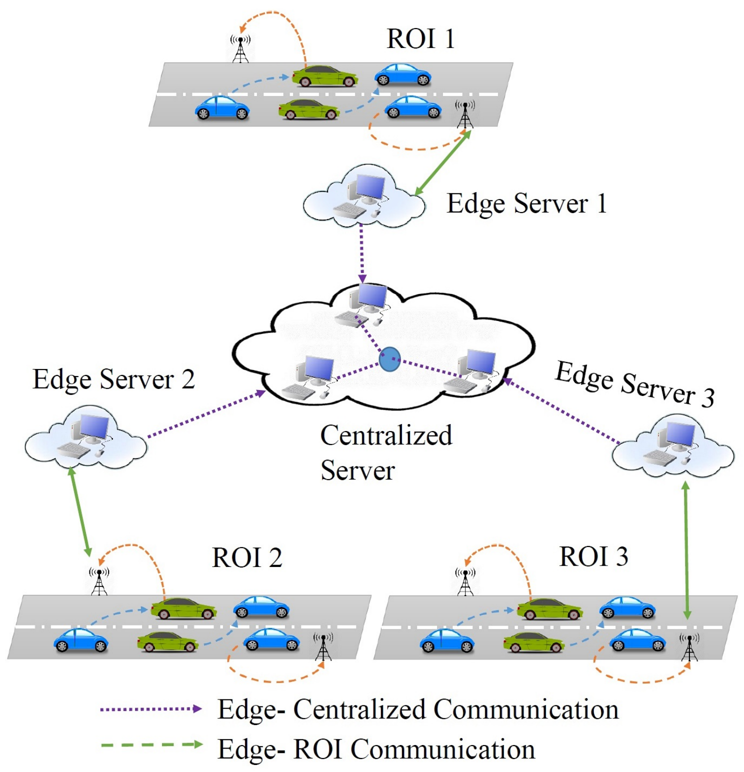


Figure 2. Proposed Model

model is not appropriate for real-time decision making such as turning the steering wheel or increasing the acceleration. It is suitable in scenarios where the requesting vehicles want to enter the ROI within approximately say *5-10 minutes* from the time of the request. In the proposed model, the RSUs are assumed to be trusted and densely populated within the ROI (such as in an urban region). The malicious vehicles can only collude with other vehicles within its transmission range.

4. OUR PROPOSED MODEL

Figure 3 represents an overview of the proposed model shown in Figure 2. It uses a privacy-preserving heuristic that leverages the GPS location and velocity of the reporting vehicle as well as encrypted neighboring V_{id} of the reporting vehicle, i.e., vehicles that are within its transmission range to validate traffic-related information at the ROI under

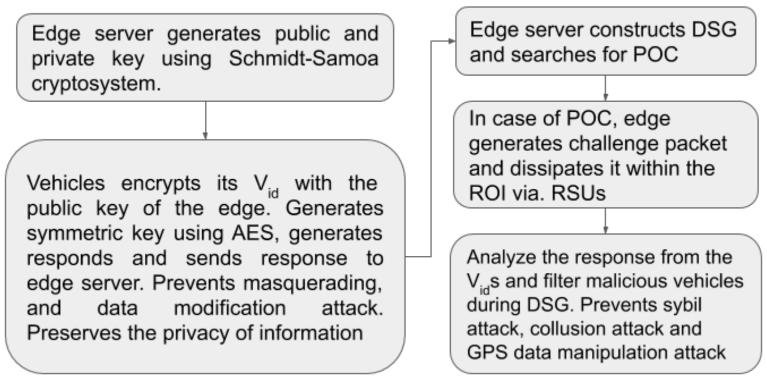

Edge server generates public and private key using Schmidt-Samoa cryptosystem.
Vehicles encrypts its V_id with the public key of the edge. Generates symmetric key using AES, generates responds and sends response to edge server. Prevents masquerading, and data modification attack. Preserves the privacy of information
Edge server constructs DSG and searches for POC
In case of POC, edge generates challenge packet and dissipates it within the ROI via. RSUs
Analyze the response from the V_ids and filter malicious vehicles during DSG. Prevents sybil attack, collusion attack and GPS data manipulation attack

Figure 3. Overview of the proposed model

the presence of at least one non-malicious vehicle. A vehicle requesting traffic-related information from an ROI sends a request to the centralized server, which is relayed to the associated edge server via a wireless communication. From one edge server, the requesting vehicle can request the traffic condition at another ROI under a different edge server. The centralized server is directly associated with all edge servers.

4.1. KEY GENERATION AND REQUEST DISSIPATION

The edge server associated with a ROI generates G_{public} and its associated $G_{private}$ using equations 1 and 2, respectively, as defined by the Schmidt-Samoa cryptosystem.

$$G_{\text{public}} = p^2 * q \tag{1}$$

$$G_{\text{private}} = {G_{\text{public}}}^{-1} \mod (lcm(p-1, q-1)) \tag{2}$$

where p and q are 2 large prime numbers chosen by the edge server. Furthermore, the edge server broadcasts G_{public} within the ROI via the RSUs.

4.2. ENCRYPTED ID BROADCAST AND RESPONSE ACQUISITION

In the proposed heuristic, every vehicle needs to know the encrypted V_{id}s of its neighboring vehicles only when generating the *data_packet*. This requirement has been justified later in this section. To preserve the privacy in the proposed model, after receiving the G_{public}, the vehicle encrypts its V_{id} using G_{public} and generates *enc_id*, which is defined as the encrypted V_{id} as discussed in Algorithm 1 (lines 6-9), and broadcasts it to its nearby vehicles.

Algorithm 1 Generate *enc_id* and τ for a vehicle

```
 1: $key[0-9, A-Z, a-z] = [00-09, 10-35, 46-71]$
 2: $Key_i \leftarrow$ Key of a vehicle generated using AES 128
 3: $Key\_m \leftarrow$ Stores integer of $Key_i$ using key[] of step 1
 4: count $\leftarrow$ length_of($\mathrm{V_{id}}$) - 1
 5: Key_length $\leftarrow$ length_of($Key_i$) - 1
 6: for i in range of $0 - length\_of(V_{\mathrm{id}}) - 1$ do
 7:     $msg = msg$ + ($100^{\mathrm{count}}$*ascii_of($\mathrm{V_{id}}$.charAt(i))
 8:     count- -
 9: $enc\_id \leftarrow msg^{\mathrm{G_{public}}}$ mod ($\mathrm{G_{public}}$)
10: for i in range of $0 - length\_of(Key_{\mathrm{i}})$ do
11:     $Keym = Keym$ + ($100^{\mathrm{Key\_length}}$*key[$\mathrm{Key_i}$.charAt(i)]
12:     Key_length- -
13: $\tau \leftarrow Keym^{\mathrm{G^{public}}}$ mod ($\mathrm{G_{public}}$)
14: return $enc\_id, \tau$
```

A vehicle having received threshold number, η, of *enc_id*s from neighboring vehicles generates *data_packet* that consists of the vehicle's information. The significance of η in the proposed model is to model the size of the *data_packet* in such a way that it consumes lesser bandwidth to send the vehicular information by the RSU.

$$data\_packet =< V_{\mathrm{id}}, ds(V_{\mathrm{id}}), event, vel_{\mathrm{id}}, GPS_{\mathrm{id}}, enc\_ids, trajectory >$$

where *ds(V_{id})* is defined as V_{id} digitally signed with *veh_private*$_i$ for authenticating itself, *enc_ids* refers to the encrypted ids of the neighboring η vehicles, *vel*$_{id}$ and *GPS*$_{id}$ are the velocity and GPS location of the vehicle respectively at the time of generating the *data_packet*, while the *event* field indicates the event recorded by the V_{id}. We do not preexamine the recorded GPS location of the vehicles and the proposed model does not deal with the precision of the GPS location of a V_{id}, as the GPS location recorded is utilized to filter malicious vehicles in the heuristic described later. The *trajectory* field describes the trajectory of the vehicle to its destination that is leveraged in section D.Thereafter, every V_{id} at the ROI generates its *Key*$_i$ and generates *encrypted_data_packet* consisting of every information that needs to be sent to the edge server in the encrypted form.

$$encrypted\_data\_packet =< \tau, data\_packet` >$$

where τ is obtained by encrypting *Key*$_i$ of a V_{id} with G_{public} (Algorithm 1 lines 10-13) and *data_packet‘* is obtained by encrypting *data_packet* with *Key*$_i$. The purpose of this step is to preserve the privacy and integrity of *Key*$_i$ and the *data_packet*. This step also facilitates the secure key exchange algorithm. Every V_{id} broadcasts the *encrypted_data_packet* received by the nearby RSU. The RSU waits for σ seconds or η number of *encrypted_data_packet*s before sending them to the edge server. The waiting time, σ, ensures that the RSU constrains the time of the proposed model in case the η packets take more time, especially in situations where the traffic flow is low. On the other hand, if the traffic flow is high, within σ time, the RSU can receive a large number of *encrypted_data_packet*s which can significantly increase the length of the aggregated packet under which it appends η *encrypted_data_packet*s obtained before σ seconds. Thus, using this trade-off, the RSU generates the *aggregate_packet* and sends it to the edge server.

$$aggregate\_packet =< rsu\_id, location, encrypted\_data\_packets >$$

where *rsu_id* and *location* are respectively the unique id and location of the RSU sending the packets to the edge.

4.3. POINT OF CONFLICT DETECTION

The edge server, on receiving the *aggregate_packet*s from the RSUs, extracts the *rsu_id* and the location of the RSU. Thereafter, it extracts the *encrypted_data_packet* from the *aggregate_packet*s received. Furthermore, from the *encrypted_data_packet*, the symmetric key, Key_i, for every vehicle is obtained by decrypting τ with $G_{private}$ using Algorithm 2 (lines 4-5). Finally, the obtained Key_i of a vehicle is further used to obtain *data_packet* from *data_packet'* (Algorithm 2 line 6). Thus, the response of a vehicle remains private from the RSU as well as from the nearby vehicles as $G_{private}$ is only possessed by the edge server. From every *data_packet*, the V_{id} is authenticated by the edge server by comparing $ds(V_{id})$ with the V_{id} and its associated veh_public_i, which it receives from the centralized server. The purpose of this step is to filter malicious vehicles that are performing any masquerading attack or identity theft.

At first, the GPS location of a $\mathrm{V_{id}}$ is compared with the location of the RSU. If the GPS location is out of the transmission range of the RSU that records its data, this implies that the vehicle is manipulating its GPS location and it is filtered out as malicious. If not, the neighbors of a V_{id} are obtained by the edge server using Algorithm 2 (lines 8-9). Based on the neighbors extracted, the edge server constructs a DSG where every V_{id} forms a vertex of the DSG, and it has an undirected edge to its neighboring V_{id}s. The undirected DSG is used to obtain the Point of Conflict (POC), defined as a situation where two neighboring vehicles, say V_i and V_j, report a conflicting event like congestion and no-congestion, respectively within the same ROI (Algorithm 2 lines 15-17) and under the same RSU. The initial detection for *POC* is searched within the neighbors using Algorithm

2 (lines 12-14). If no *POC* is detected after the initial detection, a *POC* can still exist among the vehicles within the same RSU (Algorithm 2 (lines 15-18)) if a malicious V_{id} intentionally chooses only malicious neighboring V_{id}s as analyzed in Property 1.

Consider Veh_list$_m$ = $\{V_m{}^1, V_m{}^2, V_m{}^3, ...V_m{}^k\}$ to be a finite set of malicious V_{id}, denoted by $V_m{}^i$, whereas Veh_list$_{nm}$ = $\{V_{nm}{}^1, V_{nm}{}^2, V_{nm}{}^3, ...V_{nm}{}^p\}$ is a finite set of non-malicious V_{id}, denoted by $V_{nm}{}^i$. The cardinality of a set S is denoted by $|S|$. $\Gamma(V_m{}^i)$ be a set of $V_m{}^j$s present in the neighbor list of a $V_m{}^i$ and Veh_list$_m$ $\cap$ Veh_list$_{nm}$ = $\emptyset$, i.e, a V_{id} cannot be malicious and non-malicious at the same time.

Property 1. *Given a $V_m{}^i$, it can have only $V_m{}^j$s in its neighbor list if* $\eta < |Veh\_list_m|$.

Proof. For $\eta < |Veh\_list_m|$, consider the value of η to be $|Veh\_list_m|$-1 , i.e, the maximum allowable η under the constraint.

For a given $V_m{}^i$, $V_m{}^i \cup \Gamma(V_m{}^i) = Veh\_list_m$,

when the value of $\eta = |Veh\_list_m| - 1$, meaning that the neighbor list of a $V_m{}^i$ can include all other $V_m{}^j$ to avoid the detection of *POC*.

$V_m{}^i \cup \Gamma(V_m{}^i) \subset Veh\_list_m$,

when the value of $\eta < |Veh\_list_m| - 1$, meaning that the neighbor list of a $V_m{}^i$ can include some $V_m{}^j$ to avoid the detection of *POC*.

For $\eta > |Veh\_list_m|$, consider the value of η to be $|Veh\_list_m|$+1 , i.e, the minimum allowable η under the constraint.

$\therefore$, $Veh\_list_m \subset V_m{}^i \cup \Gamma(V_m{}^i)$

This means a $V_m{}^i$ will have a $V_{nm}{}^i$ in its neighbor list, under which the condition given below holds.

$V_m{}^i \cup \Gamma(V_m{}^i) \subset Veh\_list_m \cup Veh\_list_{nm}$ □

Thereafter, if no POC is detected at all, it considers the similar event recorded by all the vehicles to be the event of the ROI. However, if a POC is detected, the edge performs initial scrutiny based on the information obtained from the vehicles in conflict as described

Algorithm 2 Obtaining data_packet and Detecting POC

```
 1: $key[0-9, A-Z, a-z] = [00-09, 10-35, 46-71]$
 2: $dec\_id$ ← decrypted $enc\_id$, POC_detected ← false
 3: neighbor_of_$V_i$ ← neighboring vehicle of a $V_{id}$
 4: for every $\tau$ received do
 5:     $Key1_i$ ← key[$\tau^{G_{private}}$ mod (p*q)]
 6: data_packet ← decrypt $data\_packet$‘ with $Key1_i$
 7: for every $enc\_id$ in $data\_packet$ do
 8:     $dec\_id$ ← $enc\_id^{G_{private}} mod(p * q)$
 9:     neighbor_of_$V_i$ ← ascii_characters_of($dec\_id$)
10: for every data_packet obtained from $V_i$ do
11:     for every $V_j$ in neighbor_of_$V_i$ do
12:         if $V_i$.event ≠ $V_j$.event then
13:             POC_detected = true, $CV_1$ ← $V_i$, $CV_2$ ← $V_j$
14:             go to Line 19
15: for any 2 vehicles, $V_k$ and $V_m$ under same $rsu_{id}$ do
16:     if $V_k$.event ≠ $V_m$.event then
17:         POC_detected = true, $CV_1$ ← $V_k$, $CV_2$ ← $V_m$
18:         go to Line 19
19: return POC_detected, $CV_1$, $CV_2$
```

below. During the initial scrutiny, the approximate velocity of a vehicle in a congested road is considered $vel_{congested}$, while the velocity in a non-congested road is considered $vel_{n\text{-}congested}$, with an allowable difference of ϵ mph to accommodate any minor variations.

1. If *event* recorded by a vehicle, say V_1, is *congested*, and its corresponding vel_1 is greater than $vel_{congested}+\epsilon$, then V_1 is considered malicious, and the V_{id} in conflict with V_1 is considered non-malicious.

2. If *event* recorded by a vehicle, say V_1, is *non-congested*, and its corresponding vel_1 is less than $vel_{n\text{-}congested}-\epsilon$, then V_1 is considered malicious, while the V_{id} in conflict with V_1 is considered non-malicious.

Subsequently, the malicious vehicles are filtered from the network using Algorithm 4 (lines 10-12), and the decision is made based on the non-malicious vehicles. However, if no decision is made after initial scrutiny, the server generates a *challenge_pkt* obtained using Algorithm 3 (line 17), consisting of V_{id}s in conflict, and the RSUs under which the conflicting $\mathrm{V_{id}}$s are expected to appear after $time_{id}$ is calculated based on the vel_{id}s, the *trajectory* of the vehicle and the $\mathrm{GPS_{id}}$s of the V_{id}s. The purpose of the *challenge_pkt* is to authenticate the vel_{id} and GPS_{id} recorded by the vehicle and to allow the V_{id}s to prove its event recorded as it is assumed to travel with almost the same vel_{id}.

$$challenge\_pkt =< CV_1, expected\_rsu_1, time_1, CV_2, expected\_rsu_2, time_2 >$$

Algorithm 3 Generating the $challenge\_packet$

```
1:  Initialize time $t$, RSUList ← list of every rsu_id
2:  for every $\mathrm{CV_i}$ do
3:      calculated_distance ← $t * vel_{\mathrm{id}}$
4:      $\mathrm{expected\_location_i}$ ← calculated_distance + $\mathrm{GPS_{id}}$
5:      for every $rsu_{\mathrm{i}} \in RSUList$ in $trajectory$ do
6:          if $\mathrm{expected\_location_i}$ is within $\mathrm{rsu_i}$.location then
7:              if $\mathrm{CV_i}$.event = "congested" then
8:                  $time_{\mathrm{i}} = t$, $expected\_rsu_{\mathrm{i}} = rsu_{\mathrm{i}}$
9:                  go to Line 17
10:             else
11:                 $expected\_rsu_{\mathrm{i}} = rsu_{\mathrm{i}}$
12:                 for every $\mathrm{rsu_j}$ beyond $\mathrm{rsu_i}$ do
13:                     $time_{\mathrm{i}} = t$, $expected\_rsu_{\mathrm{i}} = expected\_rsu_{\mathrm{i}} + rsu_{\mathrm{j}}$
14:                 go to Line 17
15:         else
16:             $t{+}{+}$
17: challenge_packet = $\mathrm{CV_i}$+$\mathrm{time_i}$+$\mathrm{expected\_rsu_i}$/s
18: return challenge_packet
```

Based on the contents of the *challenge_pkt*s, the $expected\_rsu_1$ should obtain a response from CV_1, i.e. the ID of one of the vehicles in conflict, after time, $time_1$, while the $expected\_rsu_2$ should obtain a response from CV_2, i.e. the ID of the other vehicle in conflict, after time, $time_2$. To handle the case of overspeeding by a CV_i recording "non-congested", every RSU along the direction of CV_i, obtained from its velocity, dissipates the *challenge_pkt* (Algorithm 3 lines 12-14). This ensures that even if a CV_i passed by *expected_rsu*$_i$ before *time*$_i$, it still gets the challenge packet, as over speeding is only possible in a *non-congested road*.

4.4. CHALLENGE PACKET DISSEMINATION AND RESPONSE

On receiving a *challenge_pkt* from the edge, an RSU generates a *crypto_challenge* packet and broadcasts it after *time*$_i$. The *crypto_challenge* is generated based on the CV_i assigned to it, and its purpose is to verify the presence of a vehicle within a specific region after *time*$_i$ seconds. This is leveraged in the proposed heuristic to filter the malicious vehicles within the network. Every vehicle at the ROI sends a *crypto_response*, defined as the unique response sent by V_{id} in response to the *crypto_challenge* packet. The *crypto_challenge* packet is obtained by using bitwise manipulation (*left shift* operation) over the *XOR* cipher technique. The *XOR* cipher technique is computationally inexpensive and easy to implement. Furthermore, since every RSU dissipates the *crypto_challenge* packet exactly once in the proposed model, it is less susceptible to frequency analysis attacks and also man-in-the-middle attacks. The bitwise manipulation is to enhance the security of the *crypto_challenge* after the XOR operation.

$$crypto\_challenge = CV_{\mathrm{i}} \oplus (testing\_word << left_{\mathrm{num}})$$

$left_{\mathrm{num}} \in$ [1, length of testing_word -1]
where *testing_word* is any arbitrary word chosen by a RSU having the same length as its assigned CV_i and $left_{\mathrm{num}}$ refers to the number of left shift operations performed, which is chosen arbitrarily by the RSU.

Thereafter, upon receiving the *crypto_challenge* packet from the nearby RSU, every vehicle generates the *crypto_response* packet and broadcasts it. The purpose of the *crypto_response* packet is to validate the presence of CV_i at a specific location.

$$crypto\_response = V_{\mathrm{id}} \oplus crypto\_challenge$$

The *expected_rsu*$_i$s waits for additional σ seconds to receive the *crypto_response* packets from the V_{id}s. This is done to make reparation for a minor change in vel_{CV_i} that may occur due to any trivial circumstance that does not affect the event at the ROI. However, it is assumed that vel_{CV_i} changes by a factor of atmost ϵ that still adheres to the decision recorded by CV_i. Thereafter, the RSU compares the *crypto_response* received from every $\mathrm{V_{id}}$s with the *testing_word*. The *testing_word* only matches with a specific CV_i. The associative and commutative nature of the XOR operation facilitates the effective analysis of the *crypto_response*s obtained.

Finally, the *expected_rsu*$_i$s generates the *vehicle_search* packet, which is sent to the edge server. The *vehicle_search* packets report whether a CV_i was present within the transmission range of *expected_rsu*$_i$ within $time_{\mathrm{i}} + \sigma$.

$$vehicle\_search =< rsu_{\mathrm{id}}, CV_{\mathrm{i}}, response >$$

where *response* can be *received* indicating that a CV_i is present within the range of *expected_rsu*$_i$ within $time_{\mathrm{i}} + \sigma$, or *not received* which indicates that the vehicle was absent.

4.5. DECISION MAKING BY EDGE SERVER

The edge server, on receiving the *vehicle_search* packets from *expected_rsu*$_i$s, makes a decision about traffic conditions and filters malicious vehicles based on the heuristic depicted in Figure 4. According to the proposed heuristic:

- If *crypto_response* has been received from one CV_i and is not received from the conflicting CV_i, then the CV_i from which the *crypto_response* has been received is considered non-malicious and the conflicting CV_i is considered malicious. Consequently, all the malicious vehicles with similar events recorded as the malicious CV_i are filtered using Algorithm 4 (lines 10-12). The decision is made based on the decision of the non-malicious CV_i.

- If *crypto_response* has been received by both CV_is, then the edge server assumes that the CV_i providing *crypto_response* with low vel_i intentionally reduced its velocity to prove itself non-malicious. Under such a scenario, the decision made is "non-congested". Thereafter, every malicious vehicle is filtered using Algorithm 4 (lines 10-12).

Algorithm 4 Filtering using DSG

```
1:  CV_i ← malicious vehicle detected, stack.push(CV_i)
2:  mal_list ← malicious V_id list, mal_list.append(CV_i)
3:  nonmal_list ← non malicious V_id list
4:  filter[V_id s] = false, filter[CV_i] = true
5:  while stack not empty do
6:      CV_i=stack.pop()
7:      for every V_id in DSG do
8:          if hasEdge( V_id , CV_i) and filter[V_id]==false then
9:              stack.push(V_id)
10:             if CV_i.event==V_id.event then
11:                 if CV_i== malicious then
12:                     mal_list.append(V_id)
13:                 else
14:                     nonmal_list.append(V_id)
15:             else
16:                 nonmal_list.append(V_id)
17: return  mal_list, nonmal_list
```

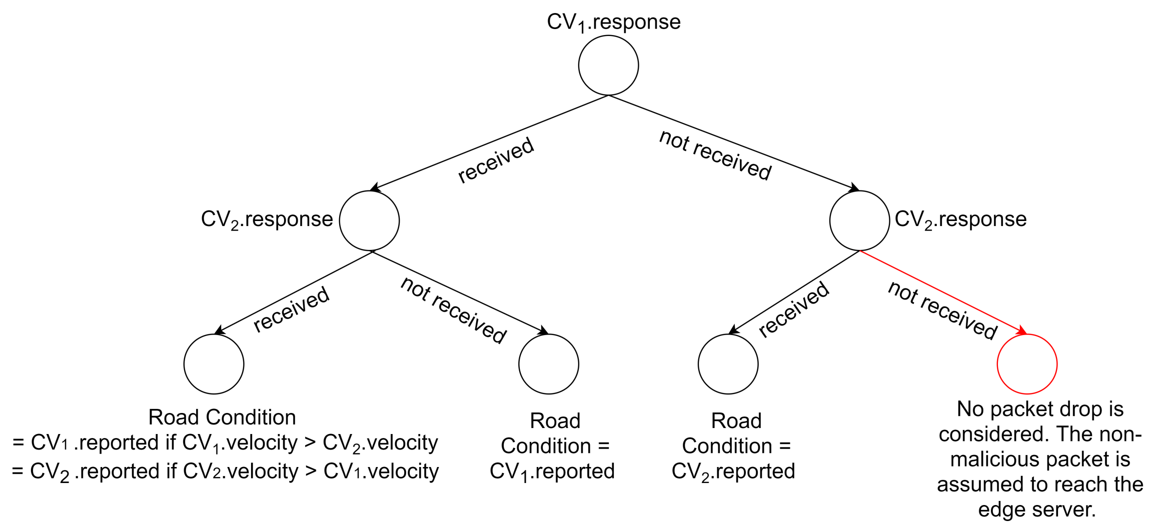

CV1.response
received
not received
CV2.response
CV2.response
received
not received
received
not received
Road Condition
= CV1 .reported if CV1.velocity > CV2.velocity
= CV2 .reported if CV2.velocity > CV1.velocity
Road Condition = CV1.reported
Road Condition = CV2.reported
No packet drop is considered. The non-malicious packet is assumed to reach the edge server.

Figure 4. Decision Tree for analysis

5. EXPERIMENTAL RESULTS AND ANALYSIS

Experiments were conducted under both congested and non-congested road conditions. The road network was simulated using SUMO [22], while the communication network was simulated using Omnet++ and VENTOS simulators. The centralized server (cloud) and the edge server were simulated in separate workstations. In our simulation, we assumed roads that have a speed limit between *70* mph and *40* mph. In other words, the roads where vehicles travel below *35* mph are assumed to be *congested* whereas the roads where vehicles travel within the speed limit are *non-congested*. Based on the experiments performed under the parameters in Table 1, it was observed that with the value of σ that were less than 10 seconds, the proposed model obtained the packets as desired. The responses of any of the CV_i did not reach the nearby RSU within time with any values of σ lesser than 10 seconds. This is because on many occasions the channel remained busy transferring packets and because of this, the *crypto_response* had to wait in the pipeline as the simulator prevents any packet collision in the channel. The value of ϵ in the proposed model can be decided based on where it is deployed and can be adjusted without any change in the performance of the model. However, for the experimentation, we considered the value to be *5* mph.

Table 1. Simulation Environment and Parameters

Consecutive RSU distance	1000 ~ 2500 meters
RSU interference distance	510.5 meters(default simulator value)
Vehicles at ROI	100
Value of σ	10
Value of *threshold*	3 vehicle data
Transmission Power	20mW (default simulator value)
Vehicle transmission range	510.5 meters (default simulator value)
Communication Protocol	IEEE 802.11p (default simulator value)
Communication channel	Dedicated Short Range Communication (default simulator value)

The proposed model has been compared against various other models using *Detection_Accuracy* as represented in Figure 5. *Detection_Accuracy* is a metric that is used to detect accurate road conditions under a varied percentage of malicious vehicles.

$$Detection\_Accuracy = \frac{road\_condition}{\%\_of\_malicious\_vehicles}$$

where *Detection_Accuracy* $\in \{1,0,0.5\}$

A value of *1* indicates that the accurate road condition and the malicious vehicles are detected while *0* indicates neither the road condition nor the malicious vehicles could be detected. A value of *0.5* indicates that the decision making is conditional, and it is either dependent on the prior reputation of vehicles (in the case of a reputation based system), the distribution of malicious vehicles (in the case of a peer authentication system) or has an equal percentage of malicious as well as non-malicious vehicles (in the case of a majority voting approach).

From Figure 5, it can be seen that the majority voting model and the peer authentication model have higher *Detection_Accuracy* when the majority of the vehicles within the ROI are non-malicious (greater than 50%). Under such cases, the decision of the non-malicious vehicles that are in the majority dominates the decision of the malicious vehicles. However, it becomes conditional when the number of malicious and non-malicious vehicles within the ROI are equal, as the non-malicious vehicles do not form a majority, and hence no proper decision could be made. Thereafter, as the number of malicious vehicles increases beyond *50%*, it is seen that the *Detection_Accuracy* decreases as the malicious vehicles form the majority under such scenarios. This influences the decision making process within the ROI. The *Detection_Accuracy* of the *Reputation_Based* model is always 0.5. This is because the decision is highly based on the reputation of the vehicles within the ROI. It is possible to have less malicious vehicles with a higher prior reputation (for instance, 5 malicious vehicles out of 100 vehicles with high rating) to dominate the majority of non-malicious vehicles with no prior reputation (for instance, 95 non-malicious vehicles out of 100 vehicles with no reputation). Thus, under such a system, the decision making model remains conditional.

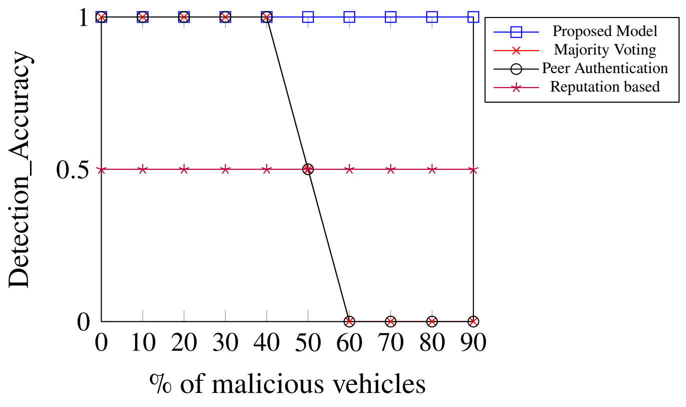


Figure 5. Comparison of Detection Accuracy of various models

It is also to be noted that the number of broadcasts required in the proposed model by the vehicles is comparatively less compared to the *peer authentication* model, as shown in Figure 6. Furthermore, we see that the peer authentication model has an equal number of broadcasts per vehicle compared to the *Proposed Model Lower*, especiallyduring the initial scrutiny phase when the *threshold* ≤ 2. However, the number of broadcasts required for decision making in the peer authentication model becomes equal to the *Proposed Model Upper* if the *challenge_response* packet is generated with $threshold = 3$, and it exceeds the *Proposed Model Upper* when the threshold becomes > 3. This is because every vehicle has to authenticate *threshold* number of vehicles in the peer authentication model, which increases with the increase in *threshold* and with the number of vehicles at the ROI. The value of *threshold* represents the number of neighboring vehicles that needs to be authenticated by a vehicle as well as the number of vehicular response sent by the RSU to the edge in one time period. Since the proposed model involves no *peer authentication*, every vehicle within the ROI has to broadcast twice during the initial scrutiny phase, and thrice if *challenge_response* is generated. However, the proposed model has more broadcast messages per vehicle compared to the *majority voting* and the *reputation-based* model. This is because the *majority voting* model and the *reputation-based* model involve no *V2V* communication, and every vehicle has to broadcast its decision only once. Furthermore, we formulate the total broadcast required for the various models shown in Table 2, where n represents the number of vehicles within the ROI and n_{rsu} represents the number of RSUs deployed within the ROI.

In the case of majority voting and reputation-based models, every vehicle within the ROI sends their response to the RSU that has a total of n transmissions. Thereafter, the RSU sends *threshold* number of vehicular responses to the edge server at one time. Hence, all the packets are sent after $\frac{n}{threshold}$ times. Therefore, the total number of transmissions required is $n+\frac{n}{threshold}$.

In the peer authentication model, every vehicle at the ROI authenticates *threshold* number of vehicles. Therefore, the number of authentication transmissions are $n * threshold$. Thereafter, the RSU sends all the packets to the edge server after $\frac{n*threshold}{threshold}$ times, i.e., n times. Thus, the total number of transmissions required in the peer authentication model is $(n * threshold) + n$, which is equivalent to $n * (1 + threshold)$.

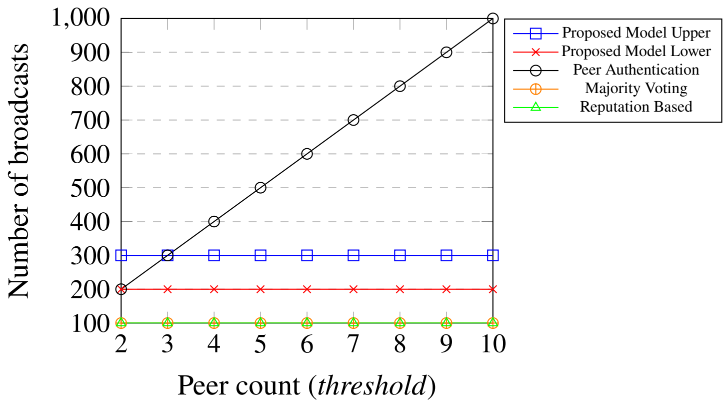


Figure 6. Broadcast comparison for 100 vehicles with varying threshold

In our proposed model, every vehicle sends its *enc_id* to its neighbor and thereafter, it sends the *encrypted_packet* to the RSU. Therefore, for every vehicle, it involves 2 transmissions. For n vehicles, the total transmissions are $2 * n$. The transmissions required by RSUs to send n packets to the edge server is $\frac{n}{threshold}$. Thereafter, the initial scrutiny is performed. If the decision is made after initial scrutiny using the proposed model, we obtain the lower bound, *Proposed Lower*, on the transmissions, which is $(2 * n) + \frac{n}{threshold}$. However, when the *challenge_response* phase is executed, the challenge packet is sent to the n_{rsu} RSUs by the edge server in one transmission, which is dissipated within the ROI. n_{rsu} RSUs broadcast the *crypto_challenge* packet and receive the *crypto_response* packets from n vehicles. Therefore, the total transmissions are $n + n_{\text{rsu}} + 1$. Finally, n_{rsu} RSUs send their response to the edge server in n_{rsu} transmissions. Under the *challenge_response* phase, we obtain the upper bound on the transmissions of the proposed model, *Proposed Upper*, which is $2 * n + \frac{n}{threshold} + 2 * n_{\text{rsu}} + n + 1$. Based on the formulation, we find from Figure 7 that the total energy required for the transmission by the majority voting and reputation-based models are the least compared to the proposed model and the peer authentication model. This is because they involve no *V2V* communication. Therefore, with fewer number of broadcast, the transmission energy used is also less. However, our proposed model has fewer number of broadcast in both initial scrutiny (Proposed Lower) and when the chal-

Table 2. Total number of broadcasts

Majority Voting	$(n+\frac{n}{threshold})$
Reputation Based	$(n+\frac{n}{threshold})$
Peer Authentication	$(n*(1+threshold))$
Proposed Lower	$(2*n+\frac{n}{threshold})$
Proposed Upper	$(2*n+\frac{n}{threshold}+2*n_{rsu}+n+1)$

lenge_response is generated (Proposed Upper) than the peer authentication model because every vehicle has a fewer broadcast requirement, i.e., two broadcasts in *Proposed Lower* and three broadcasts in *Proposed Upper*. Hence, it requires less transmission energy compared to the peer authentication model, where every vehicle has to authenticate *threshold* number of neighboring vehicles.

The time taken by our proposed model is highly dependant on the *POC* detection. It is noted from Figure 8 (plotted based on Table 3) that the detection time for our model increases with the *POC* distance (the point where two vehicles conflict in event reporting as detected by the edge). This is because only when the *POC* is detected, the edge server begins the various steps in our proposed model for detecting the road condition, and filtering out malicious vehicles. When the *POC* is detected early, i.e., within less *POC* distance, the edge proceeds with the initial scrutiny and thereafter, the *challenge_response* packet may be generated. However, with the increase in the *POC* distance, the edge server has to wait longer before executing the initial scrutiny phase. It can also be seen from Table 3 that the time taken is less dependant on the consecutive RSU distances compared to the *POC* distances during *Proposed Lower*. This is because even if the RSUs are close to each other, the edge server still has to detect a *POC* before the initial scrutiny phase is executed. The

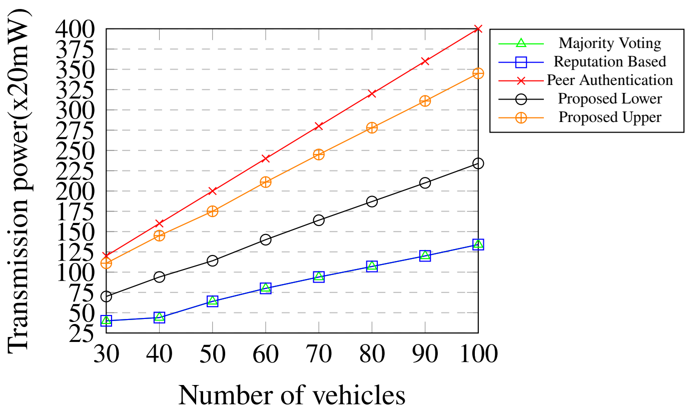


Figure 7. Comparison based on energy consumption per transmission

Proposed Upper with a consecutive RSU distance of *2500* and a *POC* distance of 10 in Table 3 takes less time (79 seconds) compared to the *Proposed Lower* with a consecutive RSU distance of *1000* and a *POC* distance of 20 (82.5 seconds). This is because in the latter scenario, the initial *POC* takes time to be detected by the edge. Therefore, a scenario having RSUs close to each other with a higher *POC* distance may take more time than a scenario with relatively distant consecutive RSUs but with less *POC* distance. Also, in the initial scrutiny phase, the time taken does not depend on the RSU distances (all the *Lower* values from Table 3 take the same time for detection for a given *POC* distance). This is because the vehicle responses is leveraged to make the decision after they are sent to the edge server. The RSU distance is impactful only when the *challenge_response* packet is generated (all the *Upper* values from Table 3 take different times for detection for a given *POC* distance).

Through experiments, it was observed that the proposed model performs faster when the vehicles remain under some RSU throughout their travel. This is because if the vehicles are out of the transmission range of the RSU, it has to wait to arrive near the next RSU before sending their corresponding packets, which increases the latency. In the experiments, every vehicle remained under the transmission range of the RSU throughout their travel when they were placed *1000* metres apart, and it was observed that *1000 Upper* as shown in Table 3 has the fastest performance when compared to *2500 Upper*, *2000 Upper*, and *1500 Upper*, i.e., when the consecutive RSU distances were *2500 metres*, *2000 metres*, and *1500 metres* apart, respectively. In Figure 9, *Upper* represents the total time taken for decision making when the *challenge_response* packet is generated, while *Lower* represents the time taken for decision making during the initial scrutiny phase.

Table 3. Time taken (in seconds) to detect traffic conditions with varying RSU and POC distances

POC Distance / RSU Distance	10	15	20	25	30	35	40
2500 Lower	21	53	84	104.0	190.3	256	326.8
2500 Upper	79	111	141	162.0	248.3	313	384.3
2000 Lower	23	54	84.33	102	193.67	254	327
2000 Upper	69	101	130.33	147	239.67	301	375
1500 Lower	22	55	82	101.7	192.5	257	327
1500 Upper	59	92	120	138.7	230.5	294.85	366
1000 Lower	24	55	82.5	103.2	194.2	257	326
1000 Upper	53	83	111.5	132.2	222.2	286	354

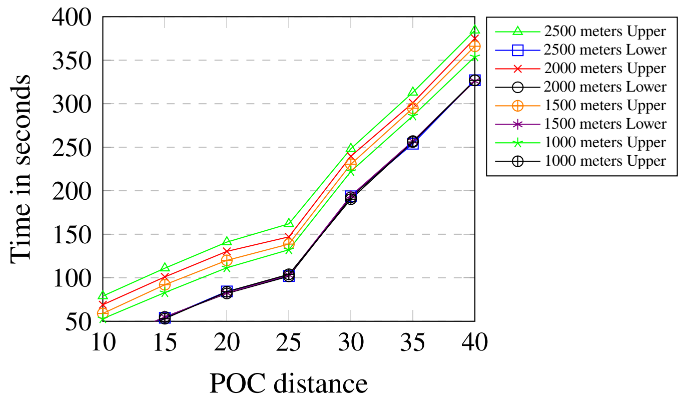


Figure 8. Broadcast comparison of 100 vehicles with varying thresholds

Table 3 also reveals the limitation of the proposed model. It can be seen that the time taken to detect the traffic condition is not in real time. This is because the amount of time lapsed in sending the *data_packet*s and detecting the traffic condition is not real time. Thus, this restricts the application of the model in scenarios that do not require real time decisions. This is justified by the fact that the ROI is usually very large, and the main purpose of the proposed model is mainly to allow the vehicles in one ROI to know the traffic scenario of another ROI that it wants to enter. However, the proposed model cannot be applied in autonomous vehicles that require decisions in negligible time, such as turning the steering wheel or braking on the road. Based on the values in Table 3, the *detection_probability* (defined as the ratio of the number of observations where the road condition is detected within a time limit to the total number of observations) is shown in Figure 9. It is to be noted that as the time increases, *detection_probability* increases. This is because with more time, conflicting vehicles at a higher *POC* distance are also considered that increase the accuracy of the model. Hence, the probability of finding a *POC* and detecting more scenarios increases as more *POC* distances are covered, and eventually, it leads to an increase in the *detection_probability* at the cost of time. For example, in Table 3, the number of observations with the time limit below 50 seconds is *4* (21,23,22,24). This means that 4 out of 56 observations (50% of the observations with *POC* distance 10) is detected within *50* seconds, thereby covering approximately *7%* of the total observations that detect the road condition with a *detection_probability* of *0.07*. However in Table 3, the number of observations detected below *100* seconds is *16*, which covers every observation with the *POC* distance of 10, *80%* of the observations with the *POC* distance of 15, and about *50%* of the observations with the *POC* distance of 20. This covers almost *28.6%* of the total observations in Table 3 (*detection_probability* is *0.28*) as compared to *7%*, when the time limit was below 50 seconds.

$$detection\_probability = \frac{total\_detections\_under\_certain\_time}{total\_possible\_detections}$$

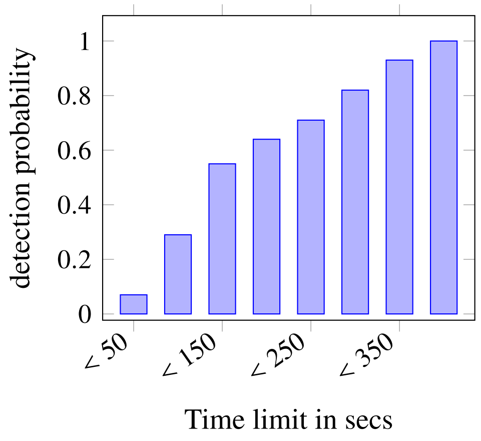


Figure 9. Detection probability with respect to time

6. SECURITY AND PRIVACY ANALYSIS OF OUR MODEL

6.1. SECURITY ANALYSIS

6.1.1. Message and GPS Information Spoofing Attack. In our proposed model, every vehicle records an *event* and generates a *data_packet*. In order to disrupt the decision making process, it can report an inaccurate event, i.e., report a congestion when the road is non-congested. For the attack to be successful, every vehicle at the ROI must spoof the event, which would incapacitate the edge server to detect any *POC*. However, this contradicts our assumption that at least one vehicle must be non-malicious. Furthermore, an attacker may spoof the *GPS* information, i.e., sent manipulated vel_{id} and GPS_{id} to the edge server. However, in the proposed heuristic, the edge server leverages vel_{id} and GPS_{id} to generate the *challenge_packet* to filter malicious vehicles within the ROI. Thus, our proposed model is secure against message and GPS information spoofing attack.

6.1.2. Masquerading, Collusion and Sybil Attack. In the proposed model, every vehicle digitally signs its V_{id} using $veh\_private_i$ generated using *EGDSS* to produce $ds(V_{id})$. An attacker can forge the signature if it can compute $veh\_private_i$ of the attacked V_{id} or by performing *hash collision attack*. However, SHA-3 [23] is secured against a collision attack, preimage [24] and second preimage attack[25] with the security strength varying from *112-256* bits for collision and *224-512* bits for preimage and second preimage attack. An attacker may also perform collusion or a sybil attack to disrupt the decision making process. However, for the attack to be successful, every V_{id}s within the ROI must be malicious. This contradicts our assumption of having one non-malicious vehicle within the ROI. Thus, the proposed model is secured against masquerading, collusion and sybil attacks.

6.1.3. Message Integrity Attack. In the proposed model, every vehicle sends $encrypted\_data\_packet = < \tau, data\_packet` >$ to an edge server. If an attacker wants to violate the integrity of a *data_packet* generated by a V_{id}, he/she must acquire $G_{private}$ stored only at the edge server, i.e. an attacker has to compute p and q used by the edger server to generate $G_{private}$. However, this violates the discrete logarithm problem [26]. An attacker may try to compute Key_i generated using AES 128 to obtain *data_packet*. However, supercomputer will take around *1 billion years* to brute-force the key [27] while the *biclique attack* [28] requires a computational complexity of $2^{126.1}$, which is highly unlikely to break in real-time. Thus, our proposed model is resilient against message integrity attack.

6.1.4. DoS Attack. Let us assume that the number of vehicles within the ROI is V_{num}.The various combinations of malicious vehicles not sending the packet to the edge server, defined as C(DoS), is given by:

$$C(DoS) = \sum_{i=1}^{V_{\text{num}}} \binom{V_{\text{num}}}{i}$$

C(DoS) represents the different number, ranging from 0 to V_{num}, of vehicles that can refrain from sending the information to the edge server via the RSU. However, *DoS* is possible only when every vehicle within the ROI drops their *data_packet*s. Thus, the probability of the *DoS* attack being successful, defined as *P(DoS)*, contradicts our assumption that every non-malicious vehicle sends their packets, at least one packet will be received by the edge server, and also that one non-malicious vehicle should be present within the ROI. Thus, the proposed model prevents *DoS* attacks.

$$P(DoS) = \frac{1}{C(DoS)}$$

6.2. PRIVACY ANALYSIS

6.2.1. Conditional Privacy Preservation. In our proposed model, every vehicle generates *enc_id* by encrypting its V_{id} with G_{public}. If an attacker attempts to track the V_{id} of a vehicle, it has to compute $G_{private}$ using the associated value of prime numbers, p and q, possessed only by the edge server. In order to breach the privacy of the *data_packet* of a vehicle, it has to obtain the Key_i, generated using AES 128, used by a V_{id} to encrypt the *data_packet*. Even under such a scenario, the attacker has to compute $G_{private}$ as Key_i is encrypted using G_{public}. Thus, the proposed model guarantees anonymity and unlinkability of a vehicle. However, the edge server filters malicious vehicles using the *DSG*. Thereafter, it is stored in the centralized server for future reference. Thus, the proposed model also guarantees conditional privacy, and only reveals the identity of the malicious vehicles when it detects a conflict.

7. CONCLUSION

Here proposed a privacy-preserving secure edge cloud-assisted traffic monitoring system for VANETs that provides accurate traffic-related information. The model is resilient against privacy attacks and unauthorized tracking, and is secured against collusion, masquerading, ballot stuffing, and bad mouthing attacks. We used *DSG* and the *challenge-response* strategy to filter malicious responses, and to determine accurate traffic-related information with fewer number of broadcasts per vehicle compared to the peer authentication model. Even though the number of broadcasts per vehicle required for the proposed model is higher than the majority voting model and the reputation based model, our model has a higher *detection_accuracy* when the number of malicious vehicles forms the majority within the ROI. This means that unlike others, our model filters malicious vehicles and accurately detects the traffic condition under the influence of at least one non-malicious vehicle. In future, we plan to extend our work using untrusted RSUs distributed sparsely throughout the ROI, such as in semi-urban and rural areas. We will also design an intrusion detection system (IDS) for the in-vehicle network to detect fabricated information injected within a vehicle without any V2X communications.

REFERENCES

[1] D Manivannan, Shafika Showkat Moni, and Sherali Zeadally. Secure authentication and privacy-preserving techniques in vehicular ad-hoc networks (vanets). *Vehicular Communications*, page 100247, 2020.

[2] Zhe Yang, Kan Yang, Lei Lei, Kan Zheng, and Victor CM Leung. Blockchain-based decentralized trust management in vehicular networks. *IEEE Internet of Things Journal*, 2018.

[3] Jun Shao, Xiaodong Lin, Rongxing Lu, and Cong Zuo. A threshold anonymous authentication protocol for vanets. *IEEE Transactions on vehicular technology*, 65(3): 1711–1720, 2015.

[4] Qin Li, Amizah Malip, Keith M Martin, Siaw-Lynn Ng, and Jie Zhang. A reputation-based announcement scheme for vanets. *IEEE Transactions on Vehicular Technology*, 61(9):4095–4108, 2012.

[5] Philippe Golle, Dan Greene, and Jessica Staddon. Detecting and correcting malicious data in vanets. In *Proceedings of the 1st ACM international workshop on Vehicular ad hoc networks*, pages 29–37. ACM, 2004.

[6] Andy Greenberg. Hackers remotely kill a jeep on the highway—with me in it. *Wired*, 7:21, 2015.

[7] William Whyte, André Weimerskirch, Virendra Kumar, and Thorsten Hehn. A security credential management system for v2v communications. In *2013 IEEE Vehicular Networking Conference*, pages 1–8. IEEE, 2013.

[8] Zhen Huang, Sushmita Ruj, Marcos A Cavenaghi, Milos Stojmenovic, and Amiya Nayak. A social network approach to trust management in vanets. *Peer-to-Peer Networking and Applications*, 7(3):229–242, 2014.

[9] Shereen A Taie and Sanaa Taha. A novel secured traffic monitoring system for vanet. In *2017 IEEE International Conference on Pervasive Computing and Communications Workshops (PerCom Workshops)*, pages 176–182. IEEE, 2017.

[10] Liehuang Zhu, Chuan Zhang, Chang Xu, Xiaojiang Du, Nadra Guizani, and Kashif Sharif. Traffic monitoring in self-organizing vanets: A privacy-preserving mechanism for speed collection and analysis. *IEEE Wireless Communications*, 26(6):18–23, 2019.

[11] Xumin Huang, Rong Yu, Jiawen Kang, and Yan Zhang. Distributed reputation management for secure and efficient vehicular edge computing and networks. *IEEE Access*, 5:25408–25420, 2017.

[12] Hamssa Hasrouny, Abed Ellatif Samhat, Carole Bassil, and Anis Laouiti. Trust model for secure group leader-based communications in vanet. *Wireless Networks*, pages 1–23, 2018.

[13] Joan Daemen and Vincent Rijmen. The block cipher rijndael. In *International Conference on Smart Card Research and Advanced Applications*, pages 277–284. Springer, 1998.

[14] Diaa Salama Abd Elminaam, Hatem Mohamed Abdual-Kader, and Mohiy Mohamed Hadhoud. Evaluating the performance of symmetric encryption algorithms. *IJ Network Security*, 10(3):216–222, 2010.

[15] Katja Schmidt-Samoa. A new rabin-type trapdoor permutation equivalent to factoring. *Electronic Notes in Theoretical Computer Science*, 157(3):79–94, 2006.

[16] Ronald L Rivest, Adi Shamir, and Leonard Adleman. A method for obtaining digital signatures and public-key cryptosystems. *Communications of the ACM*, 21(2):120–126, 1978.

[17] Michael O Rabin. Digitalized signatures and public-key functions as intractable as factorization. Technical report, Massachusetts Inst of Tech Cambridge Lab for Computer Science, 1979.

[18] Neal Koblitz. Elliptic curve cryptosystems. *Mathematics of computation*, 48(177): 203–209, 1987.

[19] Krists Magons. Applications and benefits of elliptic curve cryptography. In *SOFSEM (Student Research Forum Papers/Posters)*, pages 32–42, 2016.

[20] Taher ElGamal. A public key cryptosystem and a signature scheme based on discrete logarithms. *IEEE transactions on information theory*, 31(4):469–472, 1985.

[21] David W Kravitz. Digital signature algorithm, July 27 1993. US Patent 5,231,668.

[22] Michael Behrisch, Laura Bieker, Jakob Erdmann, and Daniel Krajzewicz. Sumo–simulation of urban mobility: an overview. In *Proceedings of SIMUL 2011, The Third International Conference on Advances in System Simulation*. ThinkMind, 2011.

[23] Morris J Dworkin. Sha-3 standard: Permutation-based hash and extendable-output functions. Technical report, 2015.

[24] Kazumaro Aoki and Yu Sasaki. Preimage attacks on one-block md4, 63-step md5 and more. In *International workshop on selected areas in cryptography*, pages 103–119. Springer, 2008.

[25] Hongbo Yu, Gaoli Wang, Guoyan Zhang, and Xiaoyun Wang. The second-preimage attack on md4. In *International Conference on Cryptology and Network Security*, pages 1–12. Springer, 2005.

[26] Kevin S McCurley. The discrete logarithm problem. In *AMS Proc. Symp. Appl. Math*, volume 42, pages 49–74, 1990.

[27] Mohit Arora. How secure is aes against brute force attacks. *EE Times*, 5(7), 2012.

[28] Andrey Bogdanov, Dmitry Khovratovich, and Christian Rechberger. Biclique cryptanalysis of the full aes. In *International Conference on the Theory and Application of Cryptology and Information Security*, pages 344–371. Springer, 2011.

II. DISTRIBUTED INCENTIVE-BASED SECURED TRAFFIC MONITORING IN VANETS

Ayan Roy and Sanjay Madria
Department of Computer Science
Missouri University of Science and Technology
Rolla, Missouri 65401
Email: ar3g3@mst.edu and madrias@mst.edu

ABSTRACT

Vehicular Ad-hoc Networks (VANETs) allow vehicles to share traffic-related events such as congestion to improve the driver's safety and comfort. However, due to the untrusted vehicular network environment, determining the credibility of broadcast messages becomes crucial and challenging. In this paper, we propose an incentive-based distributed trust management system with a secure event detection model employing the Byzantine fault-tolerant Paxos algorithm and game theory. The novelty of the proposed model lies in its ability to validate the accuracy of the broadcast information when the malicious vehicles form the majority compared to non-malicious vehicles within the ROI, unlike the state-of-the-art models. The proposed system's feasibility and effectiveness have been validated using the *VENTOS, SUMO, and Omnet*++ simulators by comprehensively addressing all possible use-case scenarios, and under the influence of at least one non-malicious vehicle at each RSU.

1. INTRODUCTION

Vehicular Ad-hoc Networks (VANETs) facilitate vehicles to share traffic-related information, such as accidents or congestion, with neighboring vehicles (termed as V2V) and with the roadside infrastructure (termed as V2I). The presence of malicious neighboring vehicles within a region of interest (ROI) can negatively influence another vehicle's accurate decision-making capabilities. In a disaster scenario, the evacuation process can be adversely affected if malicious vehicles broadcast a non-congested road as congested. Furthermore, the remote hacking of a Jeep Cherokee [1] shows that an attacker can remotely gain control of the majority of vehicles and report an inaccurate traffic event to disrupt a traffic monitoring system.

A centralized system [2] has been proposed to authenticate, store, and process the trust scores of vehicles based on their responses from the ROI. However, being a single point of attack, the system may be adversely affected if an attacker manipulates the centralized data reported by the vehicles. Also, the non-uniform connectivity of a wireless network in many locations, higher mobility of the vehicles and faster change in the network topology add to time and computational overhead.

In a distributed system, the traffic information of a region from the information shared by the vehicles is often subjected to majority voting [3], vehicle prior reputation score [2], or peer authentication [4]. However, the efficacy of such a decision-making system can be disrupted under the presence of a majority of malicious vehicles within the ROI.

To address these challenges, we propose a distributed model that overcomes the limitations of majority voting, prior reputation score, or peer authentication and provides better accuracy in determining the traffic scenario under the influence of a majority of malicious vehicles. Our model disregards prior vehicle reputation score or peer authentication, and employs an incentive-based game theoretic approaches to filter malicious information within the network.

Road Side Units (RSUs) within the ROI can be compromised by an attacker that can violate the confidentiality and integrity of the generated information. In the likelihood of such scenarios, our proposed trust model will preserve data confidentiality and integrity by adopting the Byzantine fault tolerance as defined in [5] along with the Paxos Algorithm.

Byzantine Paxos tolerates unauthorized fabrication or collusion of malicious processes. It requires a minimum of $\frac{n_{\text{process}}}{2} + 1$ uncompromised processes to deliver accurate results where $n_{process}$ is the total number of processes in the distributed system. In our model, we consider each of the RSUs deployed at the ROI to be processes of the distributed network formed among themselves. As it is difficult to compromise the majority of the RSUs in real time by an attacker [4], the use of Byzantine fault-tolerant Paxos algorithm works effectively in the proposed model to filter out malicious vehicles.

The novelty of the proposed trust model is to accurately detect malicious vehicles and the traffic scenario from the ROI based on at least one non-malicious vehicle under an RSU.

Our contributions in this work are as follows:

- Incorporated an incentive-based game theoretic approach where the vehicles, organized into clusters [6], acting as players of the game and are permitted to change the cluster, which is leveraged to filter malicious vehicles.
- Employed Byzantine fault-tolerant distributed Paxos algorithm to effectively filter compromised RSUs from the network and determine the traffic scenario of the ROI by achieving distributed consensus among the RSUs.
- Validated the effectiveness of the proposed model by simulations, with the results highlighting the accuracy and scalability of the proposed model and showing that the time taken to detect the traffic-related information from the ROI is unaffected by the presence of majority of malicious packets in the network.

2. RELATED WORK

Existing solutions mostly rely on majority voting [3], peer-based authentication [4][7] or reputation-based [2] model to monitor the traffic conditions of a region. These solutions are either centralized or distributed.

2.1. CENTRALIZED SYSTEM

In [2], the reliability of messages is determined by the centralized server based on events reported by the vehicles with high reputation scores. The model is vulnerable to ballot stuffing and bad-mouthing attacks and fails to perform when the malicious vehicles form a majority at the ROI. In [8], a micro-payment based incentive model efficiently handles packet dropping attack, and the malicious node is evicted from the network. The centralized systems suffer from bottleneck, high latency, and a single point of failure.

2.2. DISTRIBUTED SYSTEM

The decentralized schemes overcome the shortcomings of the centralized architecture using the threshold mechanism to provide authentic data from the ROI [9][10]. However, in such a system, the threshold cannot be too high or low, which otherwise drastically degrades the performance of the system. Such systems are vulnerable to ballot stuffing, bad-mouthing, sybil, and collusion attacks.

The authors in [7] have proposed a peer authentication based trust management model where the trust score is calculated either directly or indirectly. However, these peer authentication models are highly vulnerable to ballot stuffing and bad mouthing attack, in which the ratings of an individual vehicle can be influenced by nearby vehicles. The proposed model also provides incorrect traffic information in scenarios where majority of the vehicles are either malicious or compromised.

In [11], a trust model has been proposed where the vehicles generate ratings for their peers and send them to the RSUs that can be accessed by other vehicles. In [4], a weighted aggregation model utilizing the blockchain is proposed where the RSUs act as the miner nodes of the blockchain. Every vehicle calculates the credibility of a message based on the location of a vehicle and sends it to the RSU, which enters it into the blockchain. The model also assumes that the majority of the vehicles are non-malicious and is also vulnerable to ballot stuffing, bad-mouthing, and collusion attacks.

3. SYSTEM MODEL AND ASSUMPTIONS

3.1. SYSTEM MODEL REQUIREMENTS

- Trusted Authority (TA): The TA is responsible for relaying the request from a vehicle, via the RSUs, within the ROI for gathering traffic-related information, and to aid the RSU to authenticate a registered vehicle's V_{id}.

- VT_score: The $\underline{V}$ehicle$\underline{T}$rust score represents trustworthiness of a vehicle based on prior participation under the proposed system. A vehicle not participating in the decision-making process have a *null* VT_score. However, the *VT_score* of a vehicle can be +1, 0, or -1 after comparing their response with the event decided by the RSUs. A value of +1 conveys high trust value, -1 conveys low trust value, and 0 represents neutral trust value for a vehicle. Furthermore, the *VT_score* of a vehicle can be changed based on future participation.

- Υ to 50 Bait strategy: We proposed a new strategy to identify the malicious intent of a vehicle. According to the proposed strategy, 50% of the vehicles from each cluster receive Υ% of the predefined total incentive for each cluster, *incentive*, such that

$$\frac{\Upsilon}{100 - \Upsilon} > 1$$

  In this paper, the group receiving Υ% of the *incentive*, is referred to as the *Prestige* group, while the group receiving $(100 - \Upsilon)$% of the *incentive* is referred to as the *Premium* group. The need for such a constraint while distributing the incentive to the cluster members has been justified from section IV*(E)* onwards of this paper.

- Schmidt-Samoa cryptosystem [12]: The cluster heads use this cryptosystem to generate the public and private key pair. While the private keys are kept by the cluster head, the public keys are disseminated along with the respective event packets for other vehicles to encrypt their symmetric key. We prefer Schmidt-Samoa cryptosystem over RSA [13] because of its simplistic trapdoor one-way permutation. Also, unlike Rabin [14], this algorithm does not produce ambiguity in the decryption at the cost of encryption speed.

- ElGamal Digital Signature Scheme (EGDSS) [15]: The vehicles at ROI adopts the EGDSS to generate a public key, *reg_public_key*, associated with a private key, *reg_private_key*, that is stored with the TA, to authenticate itself to its cluster head. The RSU, with the aid of the TA, validates the digitally signed V_{id} to determine the authenticity of a V_{id}. Another variant of EGDSS, namely digital signature algorithm (DSA) [16], can also be used without any change in the proposed model.

- Advanced Encryption Standard (AES) [17]: Every vehicle generates its symmetric key using AES algorithm for secure key and data exchange with its cluster head. The symmetric key is encrypted using the public key generated by its cluster head as defined in [12] and is appended to its response. We prefer AES 128 bits over 192 and 256 bits because of its lower execution time [18].

3.2. THREAT MODEL

The malicious vehicles can exhibit the following behaviors:

- Message spoofing attack: A malicious vehicle can broadcast inaccurate traffic-related information.

- Denial of Service (DoS): Attackers can intentionally refrain from sending the requested information to the RSU, called a "blackhole attack", to reduce the efficacy of the proposed system. Furthermore, it can selectively drop information of some of the vehicles, called a "gray hole attack", to increase the incentive received by every other vehicle in the cluster.

- On-Off attack: A malicious vehicle can purposefully switch clusters to increase its incentive.

- Message Integrity, Masquerading, and Sybil: An attacker can impersonate another vehicle by gaining unauthorized access to the symmetric key of any nearby vehicle through data interception, perform a hash collision attack. It can perform a sybil attack and modify the data content of any broadcast packet. Message integrity attack is handled by generating keys as defined by [17] and [12], whereas Sybil and Masquerading attack are handled by implementing algorithm as defined by [15].

3.3. ASSUMPTIONS

Every malicious vehicle prioritizes its own payoff over decisions and uses greedy approach to be at the *Prestige* group of its cluster, whereas non-malicious vehicles prioritize correct decisions over incentive. Neighboring vehicles travel approximately within the same velocity range. The information broadcast by the cluster head is received by every vehicle within the transmission range. The vehicles are assumed to be registered with a *reg_public_key* and *reg_private_key* respectively, with the TA during the vehicle registration phase. However, not all registered vehicles are non-malicious. An *event* is categorized as *congested* or *non-congested*. The proposed model cannot be applied in scenarios that require real-time decision-making, such as changing lanes. It is applicable in situations when a vehicle, is approximately 4-6 minutes away from the ROI.

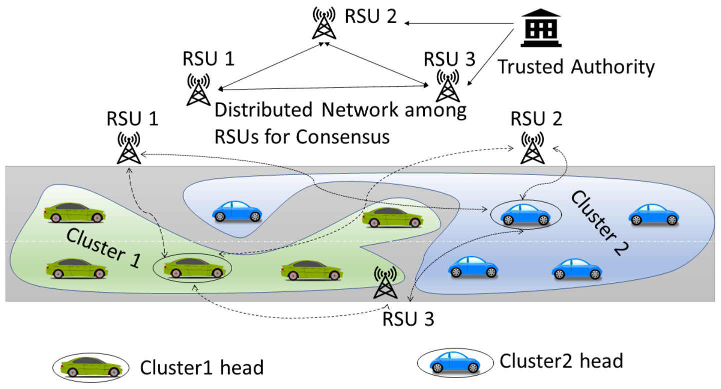


Figure 1. Proposed Model

4. PROPOSED MODEL

The main objective is to detect the accurate traffic information and filter out malicious vehicles by organizing the vehicles into clusters, leveraging the V2X communication and deploying a distributed network among the RSUs at the ROI as depicted in Figure 1. The traffic detection request is disseminated by the RSUs at the ROI. Thereafter, the various stages of the proposed model are performed as shown in Figure 2.

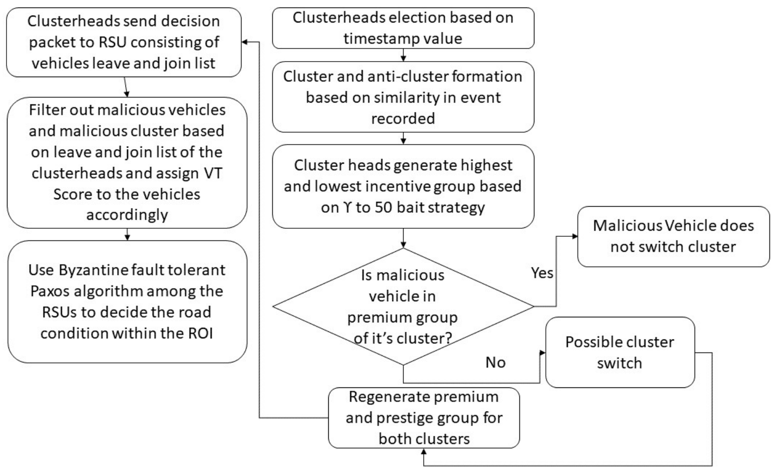

Clusterheads election based on timestamp value
Cluster and anti-cluster formation based on similarity in event recorded
Cluster heads generate highest and lowest incentive group based on ϒ to 50 bait strategy
Is malicious vehicle in premium group of it's cluster?
Yes
Malicious Vehicle does not switch cluster
No
Possible cluster switch
Regenerate premium and prestige group for both clusters
Clusterheads send decision packet to RSU consisting of vehicles leave and join list
Filter out malicious vehicles and malicious cluster based on leave and join list of the clusterheads and assign VT Score to the vehicles accordingly
Use Byzantine fault tolerant Paxos algorithm among the RSUs to decide the road condition within the ROI

Figure 2. Stages in the proposed model

4.1. CLUSTER HEAD ELECTION

In the proposed model, every vehicle recording similar events is organised in a single cluster. To form the cluster, every vehicle generates the *leader_pkt* after receiving the traffic-related event detection request from the RSU to nominate itself as the cluster head. The purpose of the *leader_pkt* is to put a V_{id} in contention for becoming the cluster head of a cluster.

$$leader\_pkt = < V_{\mathrm{id}}, ts\_val, grp\_id >$$

where *ts_val* is the timestamp value when the *leader_pkt* is generated and *grp_id* is the identifier of a cluster recording an event, which is utilized later after the cluster head is elected. A vehicle receiving a *leader_pkt* from other vehicles does not generate or drops its own *leader_pkt* if its *ts_val* is greater than the received packet. However, if it has a lower *ts_val*, it discards the received *leader_pkt* and broadcasts its own generated *leader_pkt*. After every possible *leader_pkt* from the vehicles is broadcast within a time interval of δ, the vehicle with the lowest *ts_val* is considered as the cluster head of the cluster with a unique identifier, *grp_id*. Consequently, the cluster head, named as V_{lead}, discards all other *leader_pkt*s.

4.2. EVENT PACKET DISSEMINATION BY CLUSTER HEAD

The V_{lead} generates a public key (*grp_public*) and its associated private-key (*grp_private*) using Equations 1 and 2 as defined by Schmidt-Samoa cryptosystem, and dissipates *event_pkt* consisting of cluster information to the vehicles within its transmission range. The purpose of the *event_pkt* is to to dissipate the event recorded by V_{lead} and to allow other neighboring vehicles to provide consent to the event recorded:

$$grp\_public = p^2 * q \tag{1}$$

$$grp\_private = grp\_public^{-1} \mod lcm(p-1, q-1) \tag{2}$$

$$event\_pkt =< grp\_id, event, grp\_public >$$

where p, q are two random prime numbers chosen by V_{lead}.

The V_{lead} broadcasts *event_pkt* and waits for δ seconds to receive consent to the *event* recorded by it.

4.3. CLUSTER/ANTI-CLUSTER FORMATION

Every vehicle within the transmission range of V_{lead} on receiving the *event_pkt* compares its own *event* sensed at the ROI with the *event* field of the *event_pkt* received. If it agrees to the event sensed by V_{lead}, it generates the consent packet, *consent_pkt*, to support the *event* sensed by V_{lead}.

$$consent\_pkt =< V_{\text{id}}, ds(V_{\text{id}}), grp\_id >$$

Every vehicle digitally signs its V_{id}, defined as *ds(*V_{id}*)*, with the *reg_private_key* and appends it to the *consent_pkt*. This ensures that every vehicle sends responses under its own registered V_{id}, which resists a sybil attack or masquerading attack. Thereafter, V_{id} generates *key1*, a symmetric key, using an AES 128 bit algorithm to encrypt *consent_pkt*. Finally, *data_pkt* is generated using Algorithm 5 (lines 2-11) and broadcast by V_{id}. The purpose of the *data_pkt* is to send consent to the V_{lead} from a V_{id} in an encrypted form to preserve the confidentiality and integrity of the data:

$$data\_pkt =< V'_{\text{lead}}, \tau, consent\_pkt' >$$

where V'_{lead} is the encrypted V_{lead} with *grp_public*, τ is the encrypted *key1* with *grp_public* and *consent_pkt′* is the encrypted *consent_pkt* with *key1*. Decrypting V'_{lead} only with *grp_private* facilitates V_{lead} to identify the *consent_pkt*.

In case of mismatch in the *event* sensed by a V_{id} to that of V_{lead}, V_{id} does not generate the *consent_pkt* and searches for a cluster with a cluster head recording event, contradictory to *grp_id* and similar to its own *event* sensed. If no such cluster exists, the V_{id} generates a *conflict_pkt* and broadcasts it. The purpose of the *conflict_pkt* is to form a cluster with a different traffic event contradicting *grp_id*. Eventually, V_{id} broadcasting the *conflict_pkt*, termed as $V_{antiLead}$, becomes the cluster head of the cluster, with the unique identifier as *conflict_id*:

$$conflict\_pkt =< conflict\_id, conflict\_event, anti\_grp\_public >$$

Algorithm 5 Generation of data_pkt

```
1:  $key1$ = Key by V$_{id}$, $data\_pkt \leftarrow$ "", $count \leftarrow 0$,
2:  $G_{public} \leftarrow$ $grp\_public$ for $grp\_id$, $anti\_grp\_public$ for $conflict\_id$
3:  $\tau$ = encrypted $key1$ with G$_{public}$
4:  if grp_id then
5:      $consent\_pkt'$ = encrypted $consent\_pkt$ with $key1$
6:      $V'_{cluster\ head}$= encrypted $V_{lead}$ with G$_{public}$
7:      $data\_pkt$ = {$V'_{cluster\ head}$,$\tau$,consent_pkt′}
8:  else
9:      $conflict\_consent\_pkt'$ = encrypted $conflict\_consent\_pkt$ with $key1$
10:     $V'_{cluster\ head}$= encrypted $V_{antiLead}$ with G$_{public}$
11:     $data\_pkt$ = {$V'_{cluster\ head}$,$\tau$,conflict_consent_pkt′}
12: return $data\_pkt$
```

A separate public key(*anti_grp_public*) and its private key(*anti_grp_private*), are generated using Equations 1 and 2, respectively, with different values of p and q by $V_{antiLead}$. However, in case of an existing $V_{antiLead}$, V_{id} generates *conflict_consent_pkt* instead of *consent_pkt* and *data_pkt* using its own *key1* and *conflict_consent_pkt'* by encrypting *conflict_consent_pkt* with its *key1*, following Algorithm 5 (lines 2-11) to support the event sensed by $V_{antiLead}$:

$$conflict\_consent\_pkt =< V_{\text{id}}, ds(V_{\text{id}}), conflict\_id >$$

It is to be noted that either V_{lead} or $V_{antiLead}$ can be malicious. A non-malicious V_{id} generates *conflict_consent_pkt* or *consent_pkt* based on the event sensed. On the other hand, a malicious V_{id} may sense the accurate traffic event. However, to disrupt the decision-making system, the malicious V_{id} will manipulate the event sensed and generate *conflict_consent_pkt* or *consent_pkt*, contradicting the actual traffic event.

4.4. INCENTIVE ID GENERATION

The V_{lead} and $V_{antiLead}$, on receiving the encrypted *consent_pkt* and *anti_consent_pkt*, respectively, retrieve the responses of the $V_{id}s$ using Algorithm 6 (lines 3-9) and enter the associated V_{id} and their corresponding *ds*(V_{id}) to its list of cluster members obtained using Algorithm 6 (line 10). Thereafter, V_{lead} and $V_{antiLead}$ employ Υ *to 50 Bait strategy* after which they randomly select 50% of the cluster members for receiving Υ% of *incentive* and form the *Prestige* group. Any vehicle that does not fall under the Υ% *group* is put into *Premium* group. A packet containing the V_{id}s of the *Prestige* group, defined as *Prestige packet*, is broadcast by V_{lead} and $V_{antiLead}$, and thereafter, they wait for δ seconds for any vehicle to leave or send a join request to its cluster.

4.5. INCENTIVE CALCULATION AND CLUSTER SWITCH

A malicious V_{id} on receiving the *Prestige* group packet from its cluster head examines whether its V_{id} is included in the packet or not. If not, to determine the chance of receiving a higher incentive by switching the cluster, it calculates the riskFactor(rF) using Equation 3. Since the priority of a malicious vehicle is to receive a higher incentive, it always wants

Algorithm 6 data_pkt retrieval

```
1: $V_{\text{cluster head}} = V_{lead}$ for grp_id, $V_{antiLead}$ for conflict_id
2: $G_{\text{private}} \leftarrow$ grp_private for grp_id, anti_grp_private for conflict_id
3: $V_{\text{obtained}}$ = decrypt $V'_{cluster\ head}$ with $G_{\text{private}}$
4: if $V_{\text{obtained}} == V_{\text{cluster head}}$ then
5:     Decrypt key1 from $\tau$ using $G_{\text{private}}$
6:     if $V_{\text{cluster head}} == V_{\text{lead}}$ then
7:         Decrypt consent_pkt' using key1
8:     else
9:         Decrypt conflict_consent_pkt' using key1
10: return $V_{id}$, $ds(V_{id})$, consent_pkt'/conflict_consent_pkt'
```

to be considered for the *Prestige* group of a cluster that receives $\Upsilon\%$ of the entire incentive, rather than the *Premium* group that receives $(100 - \Upsilon)\%$, where $\Upsilon > (100 - \Upsilon)$.

$$rF = (\frac{H1 - L1}{L1 \sim L2}), rF \in [-\infty, +\infty], \tag{3}$$

$$rF = \begin{cases} \text{Send Join request} & \text{if } rF \geq 1 \\ \text{Do not send join request} & \text{if } rF < 1 \end{cases}$$

Based on Equation 3, a malicious vehicle calculates the incentive that is received by an individual vehicle in *Prestige* group of the other cluster, *H1*, when all requesting V_{id}s switch, its own existing incentive, *L1*, and the incentive received in the *Premium* group of the other cluster, *L2*, after switching every requesting $V_{id}s$. *rF* value determines the involved risk of getting low incentive on switching cluster. In Equation 3, $(H1 - L1)$ determines the increase in payoff for a vehicle when moving from the *Premium* group of its existing cluster to the *Prestige* group of the other cluster, while $(L1 \sim L2)$ determines the change in payoff (increase or decrease) while moving from the *Premium* group of its existing cluster to the *Premium* group of the other cluster. The threshold of the *rF* value for the switching group is assumed to be 1 for every malicious vehicle. An *rF* value greater than 1 indicates that there is an increase of payoff, while a value equal to 1 indicates that both clusters have the same members. However, an *rF* value less than 1 indicates the presence of more vehicles in the other cluster than in the malicious $V_{id}s$ existing cluster, which eventually leads to lesser payoff.

In contrast, a non-malicious vehicle does not switch cluster even if its V_{id} is not in the *Prestige* group packet broadcast by its cluster head. When switching from a cluster, a malicious V_{id} broadcasts a *leave_join* packet where the first field of the packet indicates its existing cluster ID whereas the second field indicates the requesting cluster ID. Figure. 3 depicts a malicious V_{id}, V001, desires to switch from *grp_id* to *conflict_id*, and accordingly it broadcasts a as shown by *leave_join(V001)*.

$$leave\_join(V001) =< grp\_id, conflict\_id >$$

.

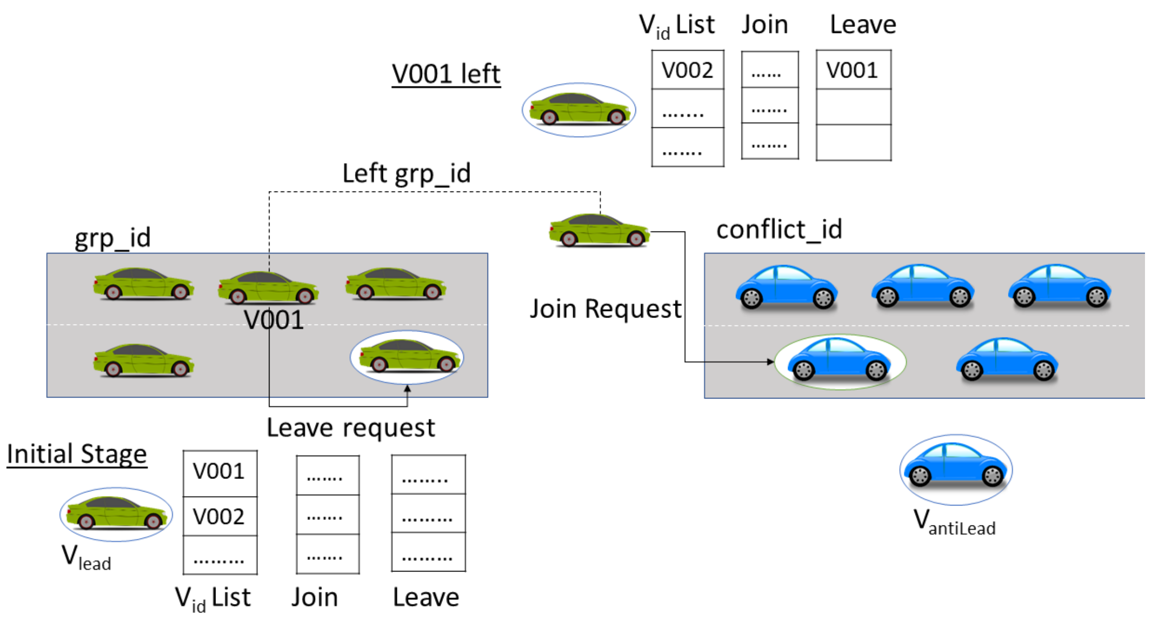


Figure 3. Switch cluster request

4.6. CONSENSUS FOR ACCEPTING NEW MEMBERS

Using game theory, the incentive is used as bait for identifying a malicious vehicle. The proposed model provides incentive to vehicles even when proved malicious. However, the *VT_score* of the vehicle decreases. This allows the model to judge whether a vehicle prioritizes its recorded event with majority support from other vehicles or its individual incentive.

The V_{lead} and V_{antiLead} on receiving the *leave_join* packet remove the V_{id}s from its cluster and enter them into its *leave_list* if its corresponding cluster ID is the first field of the *leave_join* packet. However, it generates the *join_req* packet with the requesting V_{id}s if its cluster ID is the second field of *leave_join* packet and broadcasts it to its existing cluster members. The purpose of the *join_req* packet is to notify the cluster members about the requesting $V_{id}s$ and receive consensus in regard to their acceptance into the cluster, which is a requirement of the proposed model. This determines whether the cluster members prioritize incentive over decision, that is leveraged for the decision-making by the RSUs:

$$join\_req = < v_1, v_2, v_3, ... >$$

A non-malicious V_{id} sends an *accept* message encrypted with its own *key1* to its cluster head on receiving *join_req* packet in order to substantiate the correct decision with majority vehicles. However, a malicious V_{id} calculates its probability of being in the regenerated *Prestige* group if all requesting $V_{id}s$ are accepted or rejected, represented by P(ifAccept) and P(ifReject), respectively, using Equations 4 and 5. The malicious $V_{id}s$ compromises *VT_score* over incentive since the *VT_score* value may be increased in future participation:

$$P(ifAccept) = \frac{2}{x+p} \;\; \forall \; x, p > 0 \tag{4}$$

$$P(ifReject) = \frac{2}{x} \;\; \forall \; x > 0 \tag{5}$$

where x represents the number of existing $V_{id}s$ in the cluster and p defines the number of $V_{id}s$ requesting to join.

Also, there is a possibility for the existing *Prestige* group V_{id}s of a cluster to be in the *Premium* group in the next random V_{id} regeneration for the *Prestige* group. Therefore, a malicious V_{id} calculates the incentive that would be received by an individual vehicle in the *Prestige* group and *Premium* group of the cluster if the join request is accepted, given by *H_incentive_A* and *L_incentive_A*, respectively, and if rejected, given by *H_incentive_R* and *L_incentive_R*, respectively:

$$H\_incentive\_R = \frac{\Upsilon * incentive}{50 * x}$$

$$L\_incentive\_R = \frac{(100 - \Upsilon) * incentive}{50 * x}$$

$$H\_incentive\_A = \frac{\Upsilon * incentive}{50 * (x + p)}$$

$$L\_incentive\_A = \frac{(100 - \Upsilon) * incentive}{50 * (x + p)}$$

∀x, incentive, p>0, where *incentive* is the total incentive amount. The malicious V_{id}s further calculates *metric1*, *metric2*, and *metric3* followed by the *relevancyValue*:

$$metric1 = \begin{cases} 1 & \text{if } P(ifAccept) \geq P(ifReject) \\ 0 & \text{if } otherwise \end{cases}$$

$$metric2 = \begin{cases} 1 & \text{if } H\_incentive\_A \geq H\_incentive\_R \\ 0 & \text{if } otherwise \end{cases}$$

$$metric3 = \begin{cases} 1 & \text{if } L\_incentive\_A \geq L\_incentive\_R \\ 0 & \text{if } otherwise \end{cases}$$

$$\therefore, relevancyValue = \frac{metric1 + metric2 + metric3}{3}$$

where $0 \leq$ relevancyValue ≤ 1

The *metric1* determines the favourability of a malicious vehicle to be in the *Prestige* group if the cluster head accepts requesting $V_{id}s$, *metric2* compares the individual incentive received by vehicles in the *Prestige* group by accepting requesting V_{id}s against rejecting them, and *metric3* compares the individual incentive received by vehicles in the *Premium* group by accepting new vehicles against rejecting them. The *relevancyValue* determines the reduction in incentive by accepting requesting V_{id}s and a value greater than *0.5* indicates no cutback in existing incentive for the malicious V_{id}. Only under such circumstances, it sends *agree*; otherwise, it sends *disagree* to its cluster head encrypted with its own *key1*. If malicious V_{id}s did not receive incentive even if proved malicious, then under every scenario, they would provide *agree* and it would have become challenging to filter out the malicious vehicles forming the malicious cluster. Thus, this justifies providing incentive as bait to the malicious vehicle.

A malicious V_{id} is restricted from providing *agree* to a selected requesting V_{id} because it can only send *agree* or *disagree* to the entire joining list. Even if the malicious cluster head drops packets of requesting V_{id}s from its *join_list*, it is reflected on comparing with the *leave_list* of the non-malicious cluster head during the decision making by the RSU. The use of *relevancyValue* also actuates the malicious vehicle to join the malicious cluster during the cluster formation in the beginning. This is because as the total number of vehicles changes and cannot be predicted earlier, a vehicle maliciously joining a non-malicious cluster with more members may not switch to a malicious cluster with less members. This will lead to a loss in the incentive of the malicious vehicle. Since the reverse situation is possible, it justifies the action of a malicious vehicle to join a malicious cluster at the beginning.

The cluster heads on receiving *agree* or *disagree* regenerate the *Prestige* group packet with its cluster members and thereby generates the *decision_pkt*.

$$decision\_pkt = < grp\_id/conflict\_id, HI\_members, LI\_members,$$
$$ds(V_{\text{id}})\_list, join\_list, leave\_list >$$

where *HI_members* and *LI_members* are the vehicles in the *Prestige* group and *Premium* group of a cluster respectively. $ds(V_{id})\_list$ is the digital signature of the $V_{id}s$ in a cluster.

The cluster can be switched at most once for every vehicle before reaching an equilibrium which has been proved in section V. The V_{lead} and $V_{antiLead}$, further generates their own symmetric key, *key2*, using an AES 128 bit algorithm and encrypts it with the *reg_private_key*. Thereafter, it generates the *final_pkt* and sends it to the RSU:

$$final\_pkt =< V_{\text{lead}}/V_{\text{antiLead}}, \eta, decision\_pkt' >$$

where η is defined as *key2* encrypted with *reg_private_key*, and *decision_pkt′* is *decision_pkt* encrypted with *key2*. The purpose of keeping $V_{lead}/V_{antiLead}$ unencrypted is so that during authorization, the RSU can obtain *reg_public_key* corresponding to *reg_private_key* that is stored only by the TA.

4.7. DECISION-MAKING BY RSU

On receiving the *final_pkt* from V_{lead} and $V_{antiLead}$, the RSU, with the help of the TA, obtains *key2* by decrypting η with its respective *reg_public_key*, thus verifying the authenticity of the cluster heads. An RSU authenticates every $V_{id}s$ from the *decision_pkt*, obtained by decrypting *decision_pkt'* with *key2*, by comparing the V_{id} and their associated *reg_public_key* with $ds(V_{id})\_list$ to avoid sybil and masquerading attacks. On analyzing the *join_list* and *leave_list* appended by V_{lead} and $V_{antiLead}$, the decision is based on the cluster with maximum $V_{id}s$ if the lists are empty for both *grp_id* and *conflict_id*. For other cases, the RSU adopts the strategy as represented in Figure. 4. According to Figure. 4, an RSU filters out $V_{id}s$, V_{lead}, and $V_{antiLead}$ as malicious if they do not respond using their registered *reg_private_keys*. Thereafter, an RSU checks whether any vehicle/s ($V_k s$) has changed its cluster (*Cluster1*) by examining the *leave_list* of *Cluster1*. If so, it sees whether the $V_k s$ is included in the *join_list* of the other cluster (*Cluster2*). If it is present, *Cluster2* is considered non-malicious and *Cluster1* is considered malicious. Otherwise, *Cluster2* is considered malicious and *Cluster1* is considered non-malicious.

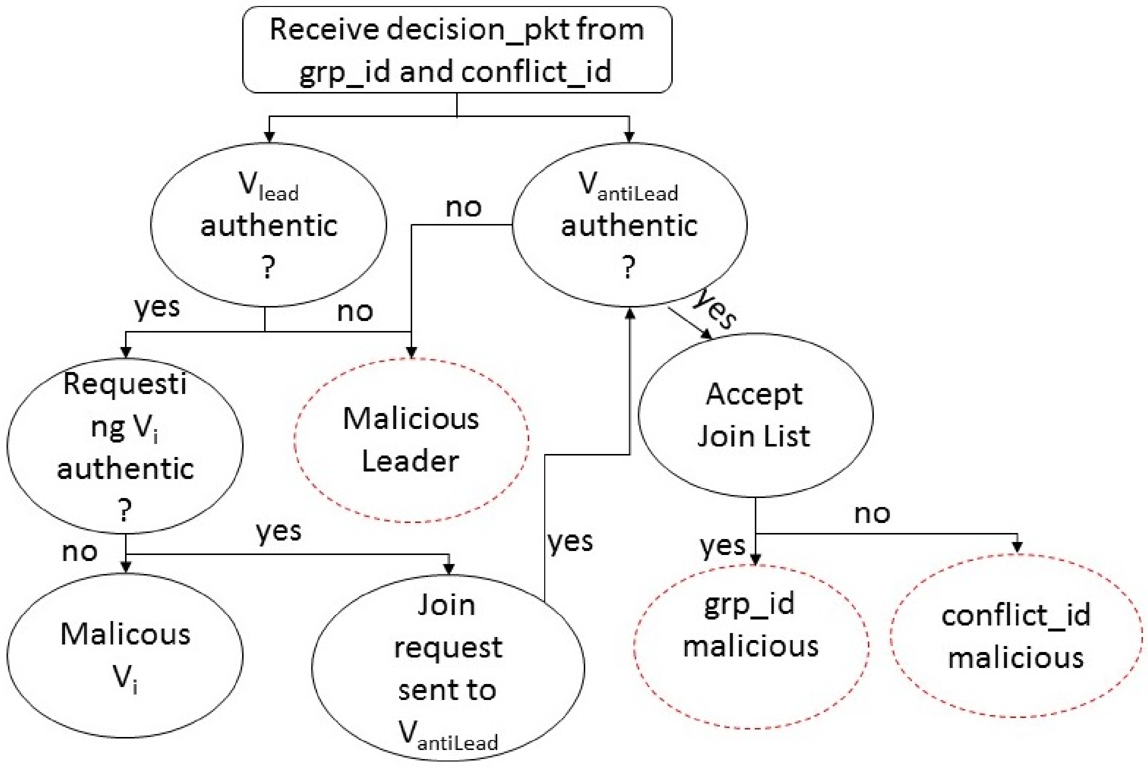
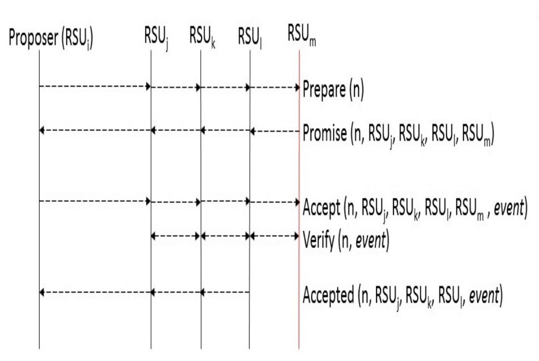

Receive decision_pkt from grp_id and conflict_id
V_{lead} authentic ?
$V_{antiLead}$ authentic ?
no
yes
no
yes
Requesting V_i authentic ?
Malicious Leader
Accept Join List
no
yes
yes
yes
no
Malicous V_i
Join request sent to $V_{antiLead}$
grp_id malicious
conflict_id malicious

Figure 4. Decision-Making by RSU

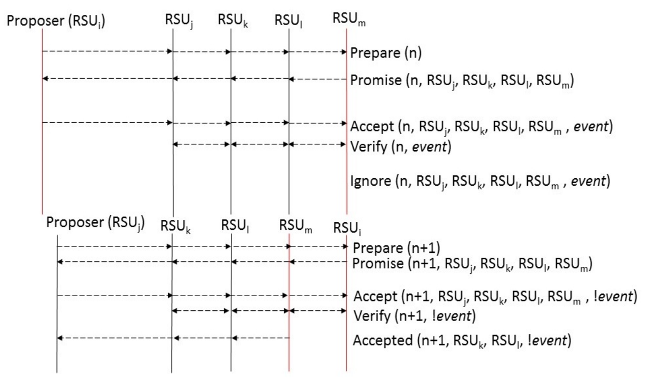
Once a decision has been made by the individual RSU at the ROI, the traffic-scenario of the entire ROI is decided by using Byzantine fault-tolerant Paxos. As depicted in Figure. 5, any one of the RSU, say *proposer*, sends a *Prepare* message with a unique sequence number, *n1*, to the other RSUs within the ROI. The other RSUs send a *Promise* message indicating that it accepts the request for consensus building. However, the RSUs could have ignored the *Prepare* message if they had already achieved consensus from some other *proposer* with a higher value of *n1*. Upon receiving the *Promise* message, the *proposer* sends an *Accept* message consisting of *n1* and the *event* decided by it. The purpose of the *Accept* message is to inform the other RSUs about the traffic event it observed and achieve consensus with it. At this stage, the other RSUs verify the *event* among themselves. A consensus among the RSUs can be reached only if it attains the quorum, i.e. the event is agreed upon by the majority of the RSUs.

$$quorum = event(\frac{n_{\mathrm{rsu}}}{2} + 1)$$

where n_{rsu} is the total number of RSUs deployed within the ROI. As depicted in Figure. 5(a), if the quorum is achieved, then the RSUs send an *Accepted* message consisting of *n1* and its corresponding RSU ID to the *proposer*, which means that the RSU conform to the event recorded by the *proposer*. However, if the *quorum* is not achieved as depicted in Figure. 5(b), the RSU does not send the *Accepted* message to the *proposer*. Instead, another RSU, say *proposer1*, becomes the new *proposer* and sends *Prepare* with a sequence number of *n1+1*. Thereafter, it sends the *Accept* message with a contradicting *event* for achieving consensus. This process ensures that the entire consensus protocol remains distributed, as well as maintains one *proposer* at a time as guaranteed by a unique sequence number. Furthermore, it also filters out compromised RSUs within the ROI based on the RSU ids included in the *Accepted* message (an RSU ID not included in the Accepted message is considered malicious). Once the malicious and non-malicious cluster is identified every vehicle of the malicious cluster is regarded as malicious and that of the non-malicious cluster as non-malicious. Finally, the event recorded by the non-malicious cluster head is considered as the event of the ROI. Thereafter, every RSU allocates the *VT_score* to every vehicle as shown below. The *VT_score* generated for every vehicle is sent by the *initiator* to the *TA* that is stored for future reference. A vehicle with a negative *VT_score* is given


Proposer (RSU_i)
RSU_j
RSU_k
RSU_l
RSU_m
Prepare (n)
Promise (n, RSU_j, RSU_k, RSU_l, RSU_m)
Accept (n, RSU_j, RSU_k, RSU_l, RSU_m , event)
Verify (n, event)
Accepted (n, RSU_j, RSU_k, RSU_l, event)

(a)


Proposer (RSU_i)
RSU_j
RSU_k
RSU_l
RSU_m
Prepare (n)
Promise (n, RSU_j, RSU_k, RSU_l, RSU_m)
Accept (n, RSU_j, RSU_k, RSU_l, RSU_m , event)
Verify (n, event)
Ignore (n, RSU_j, RSU_k, RSU_l, RSU_m , event)
Proposer (RSU_j)
RSU_k
RSU_l
RSU_m
RSU_i
Prepare (n+1)
Promise (n+1, RSU_j, RSU_k, RSU_l, RSU_m)
Accept (n+1, RSU_j, RSU_k, RSU_l, RSU_m , !event)
Verify (n+1, !event)
Accepted (n+1, RSU_k, RSU_l, !event)

(b)

Figure 5. Paxos algorithm with (a) uncompromised initiator and (b) compromised initiator

less priority for future participation. If for any vehicle, there are three instances where the *VT_score* of a vehicle is *-1*, then its corresponding V_{id} is barred from future participation and incentive.

$$VT\_Score = \begin{cases} 1 & \text{if } non\_malicious\_cluster \\ \text{-1} & \text{if } malicious\_cluster \\ 0 & \text{if } switch\_to\_non\_malicious\_cluster \end{cases}$$

5. NASH EQUILIBRIUM ANALYSIS

In this section, we analyze the Nash equilibrium of the proposed game. The payoff matrix represented in Table 1, which is applicable only for a malicious V_{id} when switching a cluster, is calculated based on the *rF value* presented in Equation 3, H_incentive_A, L_incentive_A, H_incentive_R and L_incentive_R. In Table 1, (Leave, Leave) can result in lower incentive for an individual V_{id} as the *rF value* can fall below 1.0 (depicted in Figure. 9) if a higher percentage of vehicles switches cluster. However, the *rF value* can be greater than or equal to 1 for the entry (Leave, Not Leave) if a lesser percentage of vehicles switches

Table 1. Payoff table

	Leave	Not Leave
Leave	H_incentive_A or L_incentive_A	H_incentive_A or L_incentive_A with p=1
Not Leave	$\frac{\Upsilon * incentive}{50*(x-p)}$ or $\frac{(100-\Upsilon)*incentive}{50*(x-p)}$	H_incentive_R or L_incentive_R

cluster. The entry for (Not Leave, Leave) can result in a higher individual incentive for a malicious V_{id} as the number of members in the existing cluster decreases. The entry for (Not Leave, Not Leave) will not cause any change in the individual incentive for a V_{id} in a cluster. However when otherwise switched, the malicious V_{id}s in the malicious cluster calculates the *relevancyValue* (which is less than 0.5) and does not allow the any V^{id} to switch the cluster. Thus, the malicious V_{id} can switch the cluster at most once (i.e from a malicious cluster to a non-malicious cluster).

6. SIMULATION AND EXPERIMENTAL RESULTS

The proposed model is evaluated using the VENTOS simulator [19], the ROI and the communication network are simulated in SUMO [20] and Omnet++ 5.3. Every experiment was performed in a Windows 7 Enterprise 64 bit, Intel(R) Xeon(R) CPU E5-1620 v2 @ 3.70 GHz workstation. We validated the proposed model under congested and non-congested road scenarios using the simulation parameters listed in Table 2.

For every scenario, we tested our model with both malicious V_{lead} as well as non-malicious V_{lead} against different attacks given the threat model, and compared with the different strategies adopted by the existing solutions, whose results are highlighted in Table 3. In Table 3, *Detected* indicates that the model is able to identify the correct event and the malicious vehicles, while *Not Detected* means it cannot identify the correct event under the influence of malicious vehicles. However, some of the existing models have *Conditional* detection, which means that the decision is based on other parameters. For example, in a reputation-based model or peer authentication model, the decision depends on the previous ratings of the malicious vehicles or the number of malicious vehicles surrounding a vehicle, respectively and hence, the decision is always *Conditional*.

We measure the time taken to send the *final_pkt* to the RSU by the cluster heads and the number of broadcasts required in the proposed model, as shown in Figure. 6, assuming the value of δ to be 10 and consecutive RSUs' distance to be 2000 meters. From Figure. 6(a), it is to be noted that the time taken by the proposed model to send the *final_pkt* to the RSU is less dependent on the number of malicious vehicles in the network. This is because the proposed model is bounded by the time after each broadcast to keep the solution in real time. Therefore, any malicious vehicle purposefully trying to delay the proposed system will not be taken into consideration for decision-making. The proposed model works almost in constant time for different number of vehicles in the network, proving that the proposed system is highly scalable for a large system. For Figure. 6(b), the number of non-malicious vehicles is *20%* of the total number of vehicles under one RSU, and it shows that the number of broadcasts is dependent not only on the number of vehicles but also on the percentage of malicious vehicles switching the cluster. This is because a higher percentage of malicious vehicles switches the cluster, more *leave_join* packets are broadcasted into the network. The communication cost is proportional to the number of packets which are broadcasted within the network across the various stages.

Table 2. Simulation Parameters

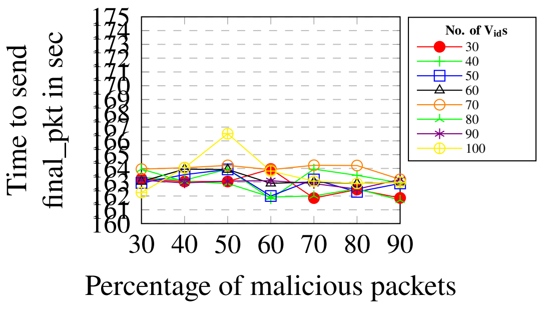
RSU Distance	800 ~ 2000 meters
RSU Interference Distance	510.5 meters (default simulator value)
Number of Vehicles at ROI	30 ~ 100
Vehicle and RSU Id Length	5
δ	5 ~ 25 seconds
incentive value	1000 units
Communication Protocol	Dedicated Short Range Communication (DSRC)

Table 3. Event Detection

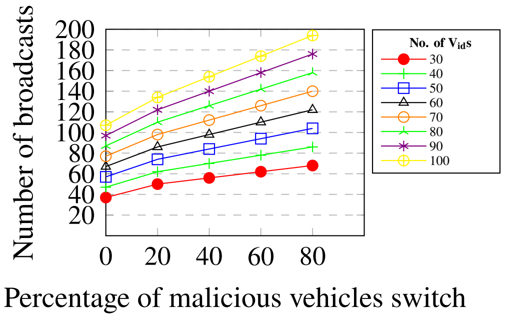
Malicious Percentage	Majority Voting	Reputation-based	Proposed Model	Peer-based Authentication
30%	Detected	Conditional	Detected	Detected
40%	Detected	Conditional	Detected	Conditional
50%	Not Detected	Conditional	Detected	Conditional
60%	Not Detected	Conditional	Detected	Conditional
70%	Not Detected	Conditional	Detected	Conditional
80%	Not Detected	Conditional	Detected	Not Detected
90%	Not Detected	Conditional	Detected	Not Detected

In Figure. 7, the value of Υ is varied to plot the incentive received by an individual V_{id} in the *Prestige* and *Premium* group of a cluster. We consider each group to consist of five vehicles. Though the value of Υ can be any value greater than 50, we kept the value of Υ to 60 because at this value, the difference in individual incentive in the *Prestige* and the *Premium* group of a cluster is half of the incentive received by a vehicle in the Premium group.

With the value of Υ as 60, in Figure 8, we plot the incentive received by an individual vehicle under the *Prestige* and the *Premium* group in the malicious and the non-malicious cluster. As the percentage of malicious vehicles in the network increases, the number of malicious packets also increases. Thus, all the malicious vehicles under such circumstances are grouped under a single cluster. For a given number of vehicles at the ROI, a growing number of malicious vehicles decreases the individual incentive at the *Prestige* and the *Premium* group of the malicious cluster, and in turn, increases the incentive at the non-malicious clusters (as seen in Figure. 8 a,b,c,d). However, in order to increase the payoff every vehicle at the ROI calculates the possibility of increasing its payoff, using the *rF value* and the payoff matrix as shown in Table 1. Thereafter, it decides whether to send a *leave_join* packet. Figure. 9 shows that as the percentage of malicious vehicles switching to a non-malicious cluster rises above *15%*, the *rF value* begins to fall below 1, which means that the malicious vehicle may incur higher loss at the Premium group of the switching cluster compared to the gain in the Prestige group. Thus, based on the *rF value*, *15%* of



(a)



(b)

Figure 6. (a) Time to send final_pkt to RSU with varying malicious packets and (b) Number of broadcasts required with varying malicious packets

malicious vehicles switching the cluster can result in an increase in broadcast by *3+n1*, where *n1* is the number of V_{id}s in the switching cluster. It is to be noted from Figure. 10 that the proposed model performs faster when the distance between consecutive RSUs at the ROI is lesser. This is because the vehicles need to wait for less time to send their data packets to the nearby RSU on account of high mobility. Based on the optimal distance of 800 meters determined from Figure. 10, we obtain the optimal value of δ in Figure. 11. It is seen that as the value of δ increases, the latency required in the proposed model also increases. This is because with an increasing value of δ, we incorporate more waiting time into the proposed model. This enables us to consider more responses from the vehicles. However, an attacker may purposefully exploit this opportunity to delay the decision making from a region. Owing to the higher mobility in vehicles, with higher values of δ, the traffic scenario at the ROI can differ drastically from the time the decision is made. For this reason, we keep the value of δ at 10 seconds. We plot Figure. 10 and 11 with 100 vehicles and 90% malicious packets.

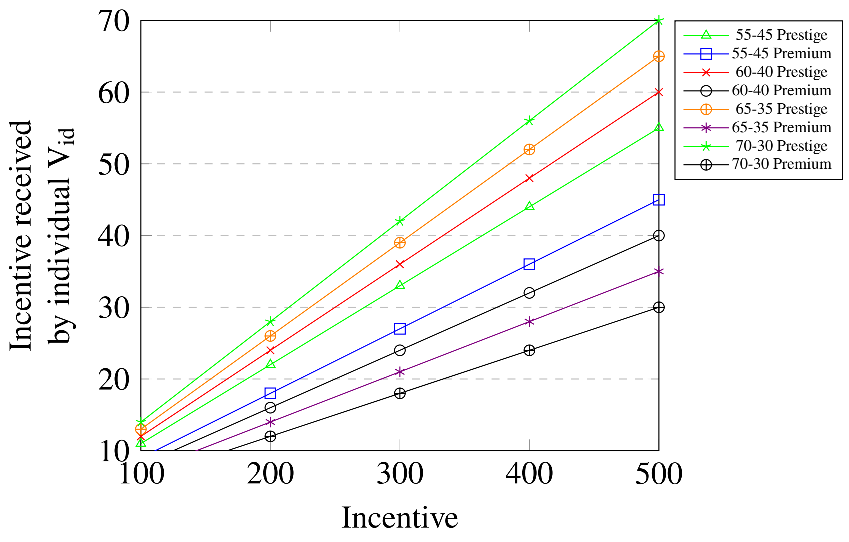


Figure 7. Υ to 50 variation

7. DISCUSSIONS ON DIFFERENT SECURITY ATTACKS

7.1. MESSAGE SPOOFING ATTACK

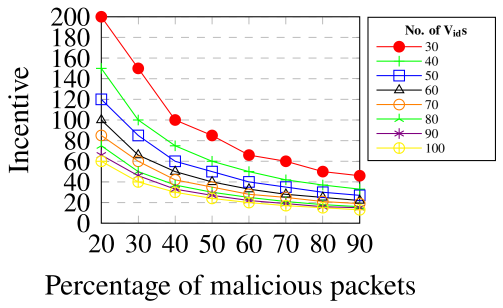
Assume the number of vehicles under a RSU is N_{num}. The malicious combinations, *f(N_{num})* is calculated by,

$$f(N_{\text{num}}) = \sum_{i=1}^{N_{\text{num}}} \binom{N_{\text{num}}}{i}$$

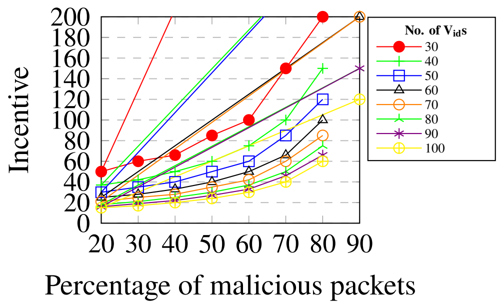
The probability of the message spoof attack to be successful, *P(Attack)*, is given by:

$$P(Attack) = 1/f(N_{\text{num}})$$

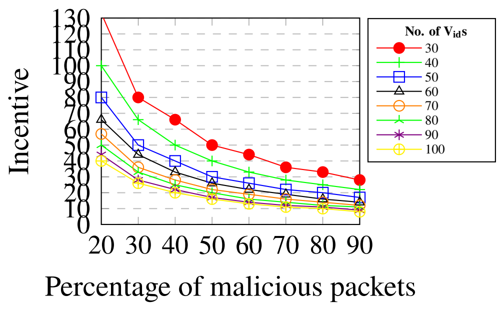
i.e., when all the vehicles are malicious. However, this contradicts the assumption of having at least one non-malicious vehicle. Hence, it is proven by contradiction that message spoofing attack is unlikely under the presence of atleast one non-malicious vehicle. A possible scenario is described below.



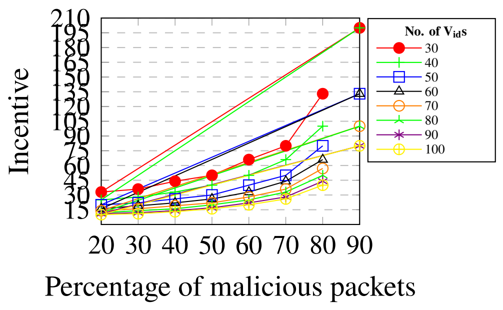
(a) Prestige group Incentive malicious cluster



(b) Prestige group Incentive non-malicious cluster



(c) Premium group Incentive malicious cluster



(d) Premium group Incentive non-malicious cluster

Figure 8. Incentives in different clusters and groups

Let us assume the vehicle count under one RSU is 20. V_{lead} is malicious and combines with 18 malicious $V_{id}s$ to form a cluster, *I001*. $V_{antiLead}$ is non-malicious and is the only vehicle of the cluster *I002* . By applying Υ *to 50 Bait strategy*, when the *Prestige* group packet is broadcast, a V_{id} in the *Premium* group of *I001* calculates the *rF* value as stated in Equation 3.

For an incentive, n, the *rF* value calculated as 1.56 > 1.

Based on *rF* value, even if one V_{id} sends *join_req*, it is accepted by $V_{antiLead}$. An RSU on receiving *final_pkt* uses the strategy in Figure. 4 and considers *I001* as malicious. Consequently, every vehicle in *I001* is considered malicious and the event of *I002* is considered as the traffic event under the RSU.

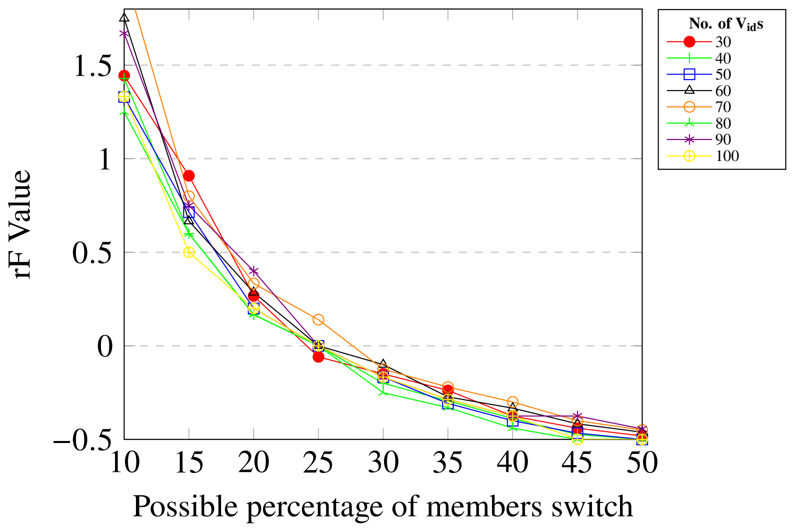


Figure 9. rF Value versus Percentage of switched members

7.2. DENIAL OF SERVICE (DOS)

The number of clusters in the proposed model, $n_{clusters} \leq 2$. Possible sample space of data reporting = $2^{\mathrm{n_{clusters}}}$. Probability of *DoS* to be successful, *P(DoS)* is given by:

$$P(DoS) = \frac{1}{2},$$

where favourable events include both malicious and non-malicious cluster head not sending *final_pkt* to RSU. Even though *P(DoS)* is high, the favourable events for *P(DoS)* contradict the non-malicious nature of a non-malicious vehicle. Hence, it is proved by contradiction that *DoS* is highly unlikely in the proposed model. The DoS attack can be performed by an attacker under the following two cases:

- Case 1: The cluster head is a malicious vehicle and does not send the *final_pkt* to the RSU.
- Case 2: The cluster head performs the gray hole attack by selectively dropping the packet of some vehicles.

Assume V_{lead} to be malicious and $V_{antiLead}$ to be non-malicious.

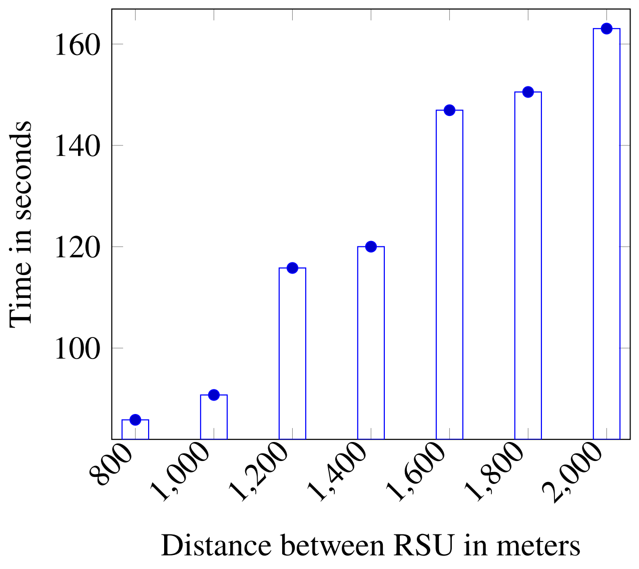


Figure 10. Optimal RSU distance in meters

For case 1, V_{lead} does not send its *final_pkt* to the RSU, while $V_{antiLead}$ sends its *final_pkt* to RSU. Under such condition, the RSU considers the road condition based on the event of the *final_pkt* from $V_{antiLead}$ only. Hence, the decision remains non-malicious and the model prevents blackhole attack.

For case 2, assume V_{lead} accepts $\{V_1,V_2,V_3,V_4,V_5\}$ for the cluster, I001, and discards the packets of $\{V_6,V_7,V_8\}$. Furthermore, V_{lead} does not broadcast the *Prestige* group packet. Upon not receiving the *Prestige* group packet from the V_{lead}, $\{V_6,V_7,V_8\}$ broadcasts a *join_req* to join the non-malicious cluster, *I002*, under $V_{antiLead}$. The RSU considers V_{lead} to drop packets if the following condition holds: if $\{\mathrm{V}_6,\mathrm{V}_7,\mathrm{V}_8\} \in$ join_list of I002 and $\notin$ leave_list of I001.

Consequently, the RSU considers V_{lead} and *I001* to be malicious and evaluates the decision and *VT_score* accordingly.

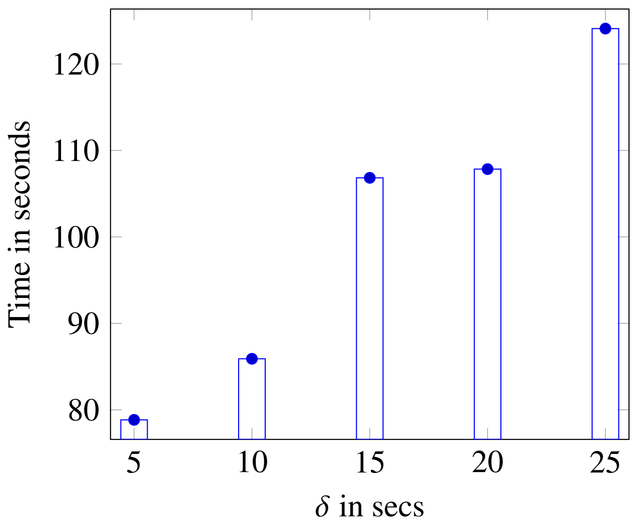


Figure 11. Optimal value of δ in seconds

7.3. ON-OFF ATTACK

Let us consider two cases of on-off attacks that are possible:

- Case 1: Possible switch from non-malicious cluster (say *I002*) to malicious cluster (say I001).
- Case 2: Possible switch from *I001* to *I002*.

For case 1, we analyze three possible scenarios based on the number of vehicles in a non-malicious cluster, non_mal_{num}, and in malicious cluster, mal_{num}:

- $\text{non\_mal}_{\text{num}} < \text{mal}_{\text{num}}$: In such cases, the malicious vehicle in *I002* calculates the *rF* value which is always less than 1. Thus, it does not leave the cluster.
- $\text{non\_mal}_{\text{num}} > \text{mal}_{\text{num}}$: In such cases, the malicious vehicles in *I001* calculate the *relevanceValue*, which while accepting vehicles, is always less than 0.5. Hence, the vehicles in *I001* do not provide consent to the requesting vehicles to switch cluster. The RSU detects the malicious group using the *leave_list* and *join_list* of the 2 clusters.

- $\text{non\_mal}_{\text{num}} = \text{mal}_{\text{num}}$: In such cases, a vehicle calculates the *rF* value proposed in the model as shown in Figure. 9 to send requests to *I001*. However, vehicles at *I001* calculate *relevancyValue* and provide consensus accordingly.

For case 2, the vehicles at *I002* being non-malicious accept every requesting V_{id}. The RSU examines the *join_list* of *I002* and makes the decision. Thus, the proposed model performs effectively against on-off attacks.

7.4. MESSAGE INTEGRITY, MASQUERADING AND SYBIL ATTACK

To forge vehicular data, one has to deduce *key1* or *key2*. The number of possible combinations in AES 128 bit is 2^{128}. A fastest supercomputer, performing around *1000 checks* per second approximately takes around *1 billion years* to brute-force the key while the *biclique attack* [21] requires a computational complexity of $2^{126.1}$, which is highly unlikely to break in real-time. Thus, using AES 128 bit, vehicular data is less likely to suffer from message integrity attack. The keys, *key1* and *key2*, are also encrypted using a [12] whose difficulty is based on discrete logarithm problem [22].

An attacker can forge the signature only if it knows *reg_private_key* of an attacked V_{id} or through *hash collision attacks*. However, SHA-3 [23] is resilient Thus, the proposed model resists masquerading, sybil, and message integrity attacks.

8. CONCLUSION

We propose an incentive based data authentication and trust management system to prevent attacks such as sybil, false data injection, message spoofing, on-off, bad-mouthing, ballot stuffing, and masquerading in VANETs. The proposed model effectively validates traffic related information under the influence of malicious vehicles, in the presence of at least one non-malicious vehicle. We also propose a trust score metric, referred to as

VT_score to determine the trustworthiness of a vehicle and introduce a filtering strategy, known as Υ *to 50 bait strategy*, that has been leveraged to filter malicious vehicles and their responses.

REFERENCES

[1] A. Greenberg, “Hackers remotely kill a jeep on the highway—with me in it,” *Wired*, vol. 7, p. 21, 2015.

[2] Q. Li, A. Malip, K. M. Martin, S.-L. Ng, and J. Zhang, “A reputation-based announcement scheme for vanets,” *IEEE Transactions on Vehicular Technology*, vol. 61, no. 9, pp. 4095–4108, 2012.

[3] P. Golle, D. Greene, and J. Staddon, “Detecting and correcting malicious data in vanets,” in *Proceedings of the 1st ACM international workshop on Vehicular ad hoc networks*, pp. 29–37, ACM, 2004.

[4] 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.

[5] M. Castro, B. Liskov, *et al.*, “Practical byzantine fault tolerance,” in *OSDI*, vol. 99, pp. 173–186, 1999.

[6] S. Gurung, A. Squicciarini, D. Lin, and O. K. Tonguz, “A moving zone based architecture for message dissemination in vanets,” in *Proceedings of the 8th International Conference on Network and Service Management*, pp. 184–188, International Federation for Information Processing, 2012.

[7] H. Hasrouny, A. E. Samhat, C. Bassil, and A. Laouiti, “Trust model for secure group leader-based communications in vanet,” *Wireless Networks*, pp. 1–23, 2018.

[8] M. E. Mahmoud and X. Shen, “An integrated stimulation and punishment mechanism for thwarting packet dropping attack in multihop wireless networks,” *IEEE Transactions on Vehicular Technology*, vol. 60, no. 8, pp. 3947–3962, 2011.

[9] J. Shao, X. Lin, R. Lu, and C. Zuo, “A threshold anonymous authentication protocol for vanets,” *IEEE Transactions on vehicular technology*, vol. 65, no. 3, pp. 1711–1720, 2015.

[10] S. A. Taie and S. Taha, “A novel secured traffic monitoring system for vanet,” in *2017 IEEE International Conference on Pervasive Computing and Communications Workshops (PerCom Workshops)*, pp. 176–182, IEEE, 2017.

[11] X. Huang, R. Yu, J. Kang, and Y. Zhang, "Distributed reputation management for secure and efficient vehicular edge computing and networks," *IEEE Access*, vol. 5, pp. 25408–25420, 2017.

[12] K. Schmidt-Samoa, "A new rabin-type trapdoor permutation equivalent to factoring," *Electronic Notes in Theoretical Computer Science*, vol. 157, no. 3, pp. 79–94, 2006.

[13] R. L. Rivest, A. Shamir, and L. Adleman, "A method for obtaining digital signatures and public-key cryptosystems," *Communications of the ACM*, vol. 21, no. 2, pp. 120–126, 1978.

[14] M. O. Rabin, "Digitalized signatures and public-key functions as intractable as factorization," tech. rep., Massachusetts Inst of Tech Cambridge Lab for Computer Science, 1979.

[15] T. ElGamal, "A public key cryptosystem and a signature scheme based on discrete logarithms," *IEEE transactions on information theory*, vol. 31, no. 4, pp. 469–472, 1985.

[16] D. W. Kravitz, "Digital signature algorithm," July 27 1993. US Patent 5,231,668.

[17] J. Daemen and V. Rijmen, "The block cipher rijndael," in *International Conference on Smart Card Research and Advanced Applications*, pp. 277–284, Springer, 1998.

[18] D. S. A. Elminaam, H. M. Abdual-Kader, and M. M. Hadhoud, "Evaluating the performance of symmetric encryption algorithms.," *IJ Network Security*, vol. 10, no. 3, pp. 216–222, 2010.

[19] M. Amoozadeh, H. Deng, C.-N. Chuah, H. M. Zhang, and D. Ghosal, "Platoon management with cooperative adaptive cruise control enabled by vanet," *Vehicular communications*, vol. 2, no. 2, pp. 110–123, 2015.

[20] P. A. Lopez, M. Behrisch, L. Bieker-Walz, J. Erdmann, Y.-P. Flötteröd, R. Hilbrich, L. Lücken, J. Rummel, P. Wagner, and E. WieBner, "Microscopic traffic simulation using sumo," in *2018 21st International Conference on Intelligent Transportation Systems (ITSC)*, pp. 2575–2582, IEEE, 2018.

[21] A. Bogdanov, D. Khovratovich, and C. Rechberger, "Biclique cryptanalysis of the full aes," in *International Conference on the Theory and Application of Cryptology and Information Security*, pp. 344–371, Springer, 2011.

[22] K. S. McCurley, "The discrete logarithm problem," in *Proc. of Symp. in Applied Math*, vol. 42, pp. 49–74, USA, 1990.

[23] M. J. Dworkin, "Sha-3 standard: Permutation-based hash and extendable-output functions," tech. rep., 2015.

III. BLAME: A BLOCKCHAIN-ASSISTED MISBEHAVIOR DETECTION AND EVENT VALIDATION IN VANETS

Ayan Roy and Sanjay Madria
Department of Computer Science
Missouri University of Science and Technology
Rolla, Missouri 65401
Email: ar3g3@mst.edu and madrias@mst.edu

ABSTRACT

Vehicular Ad-hoc Networks (VANETs) allow vehicles to share traffic-related events such as congestion to improve the driver's safety and comfort. However, due to the untrusted vehicular network environment, determining the credibility of broadcast messages becomes crucial and challenging. In this paper, we propose an incentive-based distributed trust management system with a secure event detection model employing the Byzantine fault-tolerant Paxos algorithm and game theory. The novelty of the proposed model lies in its ability to validate the accuracy of the broadcast information when the malicious vehicles form the majority compared to non-malicious vehicles within the ROI, unlike the state-of-the-art models. The proposed system's feasibility and effectiveness have been validated using the *VENTOS, SUMO, and Omnet*++ simulators by comprehensively addressing all possible use-case scenarios, and under the influence of at least one non-malicious vehicle at each RSU.

1. INTRODUCTION

Vehicular ad-hoc network (VANET) facilitates the dissemination of safety information to nearby vehicles through vehicle-to-everything (V2X) communications. The motivation for information exchange within the VANET is to ensure driver comfort and safety, as well as to assist other vehicles, such as emergency vehicles, into taking the shortest route to its destination based on the traffic event in a region. However, malicious vehicles can inject false or fabricated information within the network to disrupt the traffic scenario or a region. Owing to the connectivity of the vehicles within the network through V2X communication, an attacker can compromise a vehicle remotely such as the remote hacking of jeep Cherokee [1] and utilize it to propagate or infect other vehicles within the network. A comprehensive analysis to preserve the privacy, authenticity, and security of the messages disseminated in VANET has been given in [2]. It becomes challenging to detect the malicious vehicles and traffic conditions when the attack propagation within the network compromises the majority of the vehicles deployed within the region of interest (ROI). This situation can also occur when a malicious vehicle tries to form a majority by providing incentives to other vehicles to report.

Centralized solutions involving the use of a trusted centralized entity (TA) have been proposed [3] to thwart the impact of malicious vehicles within the network. Such a solution is widely effective in regions of continuous connectivity with the TA. However, the efficacy of these solutions degrades in a rural transportation system due to limitations in the connectivity of the vehicles to the TA at every time. Additionally, in a centralized architecture, communication with the TA may add to latency in decision making due to network congestion, and reduce the reliability of the system with the prospect of a single point of failure.

In such a case, a distributed solution adapts to the demands of secure message dissemination, and identification of malicious vehicles without a trusted centralized entity. With the popularity of the blockchain in recent years, researchers have extensively analyzed the application of blockchain for secure message dissemination within VANETs. A blockchain integrated VANET architecture consists of 3 major layers: a vehicle layer (or vehicular plane), a roadside unit (RSU) layer (or RSU plane), and a blockchain hosted by the infrastructure [4, 5, 6, 7, 8, 9] responsible for computing the trustworthiness of the vehicle and the validity of an event recorded. However, a traditional blockchain-assisted VANET cannot provide valid information from the ROI if the majority of the vehicles are compromised through attack propagation.

To address these challenges, in this paper, we have proposed a BLockchain Assisted Misbehavior detection and Event (*BLAME*) validation framework that is capable of validating event information from the ROI, even when the majority of the vehicles within the network are malicious, and if a malicious vehicle appears as a ghost vehicle. In *BLAME*, we have involved all the 3 major components as considered in traditional architecture. The RSUs are responsible for hosting and adding the blocks into the blockchain whereas the vehicles send encrypted neighbor information to the RSUs for efficient detection of an event by the RSUs. The main contribution of *BLAME* are as follows:

- First, we propose a secure and privacy-preserving misbehavior detection and event validation scheme in VANET. In our scheme, we leverage the pseudo-identity, public key infrastructure, and timestamp information to resist various attacks on the disseminated information within the network.

- Second, we propose a new algorithm called Aggregated Vehicle Density (*AVD*) and neighbor aggregation (*NeA*). The *AVD* is used to determine the event of an ROI and identify malicious vehicles within the network by analyzing the encrypted neighbor and event information received from the vehicles. The *NeA* is used to determine the aggregated neighbors of 2 vehicles with conflicting reported events.

- Third, we propose a new consensus mechanism based on the authenticity of the vehicle, the timestamp information, and the complaint information by any vehicle against any RSU to identify any compromised or malicious RSUs within the blockchain network.

- Fourth, we validate the efficacy of *BLAME* through extensive simulations by considering possible use case scenarios in the VENTOS [10] simulator and a simulated blockchain environment.

2. RELATED WORK

Singh et. al [4] proposed an incentive-based trust management scheme for event detection and validation among the vehicles using sharded blockchain network deployed within the RSUs, and by using smart contracts. The purpose of the sharded blockchain is to reduce the propagation delay and to reduce the work-loads from the main maintained blockchain. However, the proposed framework is effective for event validation only when the malicious nodes at the vehicle plane (vehicles) are in minority. The proposed framework is also vulnerable to collusion attack among the vehicles and the RSUs, the Sybil attack by the vehicles at the vehicle plane, and also data manipulation attack.

A voting blockchain-based message dissemination involving the reputation of vehicles [6] and a multihop incentive-based solution is proposed in [11] where several vehicles vote for a particular message. If the vote is above a threshold, it is considered authentic. However, the proposed model is vulnerable to ballot stuffing or bad-mouthing attacks where a group of malicious vehicles colluding can purposefully elevate the reputation of another malicious vehicle or lower the reputation of a legitimate vehicle. Also, a malicious vehicle can perform a blackhole attack and drop an endorsed packet without forwarding it.

Lu et al. [12][13] proposed a blockchain-based trust management framework (BARS) where the reputation of the vehicle was determined based on previous interactions. However, the proposed model is vulnerable in a situation where a vehicle performs an on-off attack, i.e., a vehicle with a higher reputation injects false information within the network. DrivMan [14] is a trust management framework where the reliability of the data originating from a registered source is ensured using physical unclonable functions (PUF). However, DrivMan does not deal with message validation and is not effective if a registered vehicle reports a false message to the RSUs. Nisha et al. proposed a blockchain framework [5] for identity authentication and revocation in VANETs but fails to validate information generated by the vehicles. They also proposed a secure message exchange framework in blockchain [9] where the credibility of the vehicle is determined by comparing the packet delivery rate, received signal strength indicator as well as a partnership for renewable against a threshold value. However, it cannot validate the information originating from the malicious vehicles which have the values of the mentioned parameters above the threshold. Mostafa [7] proposed a general framework for detecting malicious vehicles within VANET. In the proposed framework, the blockchain network is maintained by the vehicles at a region, and the information is validated by matching the length of the data sent by the vehicle trying to join the network. However, the proposed model is vulnerable if the existing vehicle within the network is malicious and purposefully does not allow a legitimate vehicle from joining the network.

Khelifi et al. [8] proposed a reputation-based blockchain mechanism to secure information and data forwarding as well as content caching. In the proposed blockchain framework, a look-up table is maintained to store the reputation of the authentic vehicles that are used to validate the information. However, the proposed model is vulnerable to an on-off attack where a malicious vehicle with a higher reputation injects false information or it purposefully reduces its reputation and thereafter injects correct information which is neglected using the framework. The proposed model is also vulnerable to an attack led by the collusion of malicious vehicles.

In *BLAME*, we overcome the limitations of the state-of-the-art models and ensure the security of information as well as validation of the information both at the vehicle plane (i.e., the information obtained from the ROI) as well as at the RSU plane that maintains the blockchain. Furthermore, *BLAME* can handle situations when the majority of vehicles at the vehicle plane are malicious or compromised.

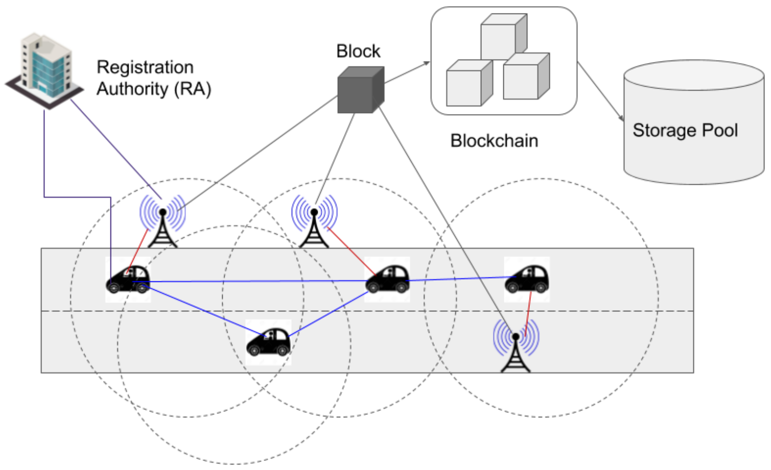


Figure 1. Architecture of *BLAME*

3. SYSTEM, ASSUMPTIONS, AND THREAT MODEL

3.1. SYSTEM ARCHITECTURE

A detailed description of the components of Figure 1 is as follows:

- Blockchain: In *BLAME*, we use the consortium blockchain that is hosted by the RSUs deployed within the ROI. The vehicles can download the updated blockchain to validate the entry of information. However, the RSUs are the only miner nodes in our proposed model. For reaching the consensus under the influence of malicious RSUs within the blockchain network, we analyze the complaint information submitted by any vehicle against any RSUs as well as the legitimacy of the vehicle submitting the information. Each block consists of the sender id, the hash of the previous block, the timestamp of the block generation, the hash of the information generated using the hash function sent by the sender, and an index to the storage pool.

- Storage pool: The storage pool maintains detailed information of the vehicle sending the information. It stores the index pointed by the blockchain, the hash of the sender, the sender id, the event reported by the sender, and the neighbor information sent by the vehicle.

- Registration Authority (RA): To ensure the legitimacy of the entities, every vehicle as well as the RSUs are registered with unique public and private key pair with the RA before entering into the VANET communication.

- Elgamal Digital Signature Scheme (EGDSS) [15]: The public and private key parameters for every vehicle that are registered with the RA is generated using the *EGDSS* as explained in section IV.A.

- Elgamal Encryption System [15]: The RSUs within the ROI generate a cyclic group, G of order q with a generator, $\mathscr{G}$. An RSU within the ROI chooses a secret integer, $\mathscr{G}_{\text{private}} \in \{1, ..., q-1\}$, shares it with all other RSUs within the ROI, and calculates, $\mathscr{G}_{\text{public}} = (G, \mathscr{G}, q, r)$, where $r = \mathscr{G}^{\mathscr{G}_{\text{private}}}$.

- Aggregated Vehicle Density (AVD): We proposed a new algorithm, *AVD*, to determine the traffic event of the ROI and also to determine the malicious vehicles within the network. To determine AVD, we define a parameter called *neighbor aggregation (NeA)*, which is calculated between a pair of vehicles reporting conflicting events or present within the same location, as follows:

  Let V_m^i be the neighbor list reported by a vehicle V_i. For every vehicle V_j in V_m^i reporting conflicting events, the NeA is calculated as:

$$NeA(V_i, V_j) = true$$

  From *NeA*, the *AVD* is calculated as follows:

$$AVD_k = \begin{cases} AVD_{k\text{-}1} \bigcup (V_i \bigcup V_j), & V_i, V_j \notin AVD_{k\text{-}1} \\ AVD_{k\text{-}1} \bigcup V_m^i, & V_i \notin AVD_{k\text{-}1}, \\ AVD_{k\text{-}1} \bigcup V_m^j, & V_j \notin AVD_{k\text{-}1}, \\ AVD_{k\text{-}1}, & V_i, V_j \in AVD_{k\text{-}1} \end{cases}$$

where *AVD* consists of the vehicle ids which are considered for calculating the vehicle density of the ROI. The *AVD* is calculated using incremental approach, i.e., *AVD* at kth comparison is calculated using the *AVD* at (k-1)th comparison.

3.2. ASSUMPTIONS

The assumptions in *BLAME* are as follows: 1) Majority of the RSUs are trusted entities. 2) The information is shared by the vehicles and the RSUs with the RA through a secured channel. 3) The proposed model cannot be used for real-time decision-making such as lane changing. 4) The packets from the non-malicious vehicles (even if they form the minority) must reach the ROI for accurate event detection and malicious vehicle identification. 5) The traffic event considered in our work can either be congested or non-congested. 6) The velocity log cannot be modified by the vehicles. 7) The safety distance is uniform and is generally fixed by the department of transportation. 8) The transmission range of the vehicles within the ROI is uniform.

3.3. THREAT MODEL

The malicious vehicles within the ROI can do the following: 1) inject false information within the network by manipulating the event recorded by it. 2) drop the packet of the nearby non-malicious vehicles. 3) replay the encrypted neighbor packet of one vehicle to other vehicles. 4) collude with other malicious vehicles (to form a majority) within the ROI to disrupt the traffic monitoring of the ROI.

The malicious RSUs within the ROI can do the following: 1) drop the information sent by the non-malicious vehicle 2) modify the content of the information reported by the malicious vehicle. 3) add false information about the non-malicious vehicles into the storage pool.

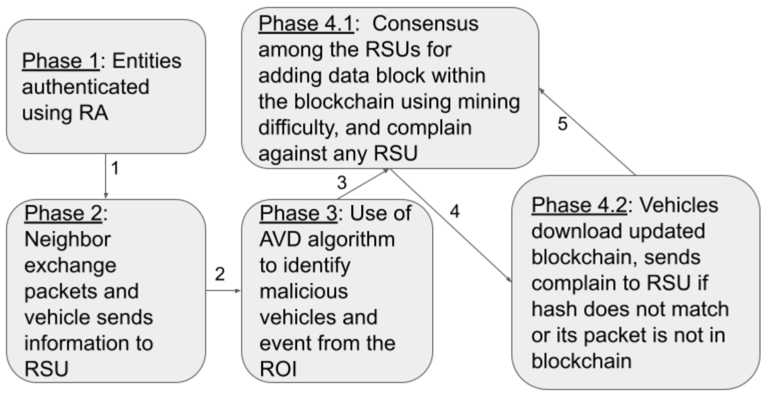

Phase 1: Entities authenticated using RA
1
Phase 2: Neighbor exchange packets and vehicle sends information to RSU
2
Phase 3: Use of AVD algorithm to identify malicious vehicles and event from the ROI
3
Phase 4.1: Consensus among the RSUs for adding data block within the blockchain using mining difficulty, and complain against any RSU
4
Phase 4.2: Vehicles download updated blockchain, sends complain to RSU if hash does not match or its packet is not in blockchain
5

Figure 2. Overview of *BLAME*

4. PROPOSED BLAME MODEL

In *BLAME*, we deploy a distributed blockchain network among the RSUs to identify the traffic event from the ROI under the influence of malicious vehicles and RSUs by following the steps shown in Figure 2.

4.1. REGISTRATION WITH RA

Every V_{id} must be registered with the RA for VANET communication. The registration takes the following steps:

- A V_{id} chooses a large prime, $\mathscr{P}$, a hash function, $\mathscr{H}$ and a primitive root, α, modulo $\mathscr{P}$ such that:

$$\alpha^{\mathrm{k}} \equiv \mathscr{A}(mod\ \ \mathscr{P}) \tag{1}$$

  where $\mathscr{A}$ is relatively prime to $\mathscr{P}$ with value between $1-(\mathscr{P}-1)$, i.e., $\alpha \in \mathbb{Z}^*{}_{\mathscr{P}}$, and $\mathrm{k} \in \mathbb{Z}$.

- The V_{id} chooses a secret key, $\mathscr{X}$, and calculates :

$$\beta \equiv \alpha^{\mathscr{X}}(mod\ \ \mathscr{P}) \tag{2}$$

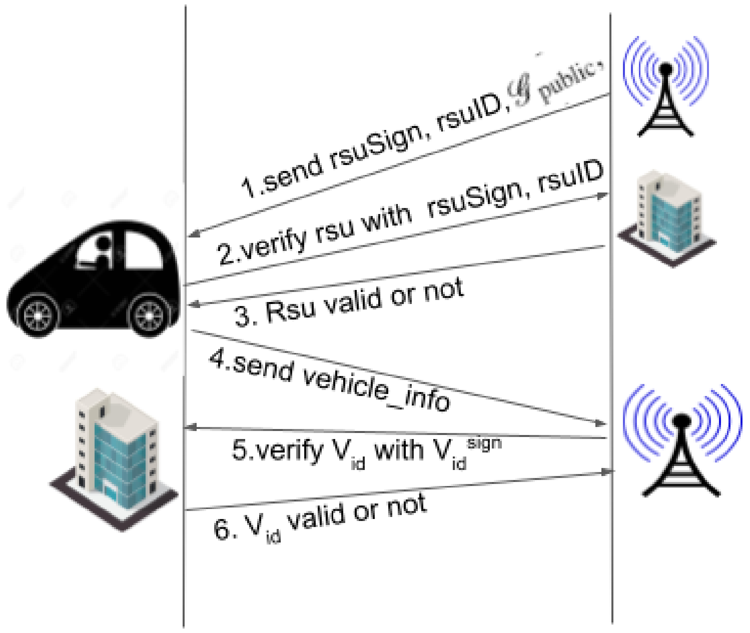

1.send rsuSign, rsuID, 𝒢 public,
2.verify rsu with rsuSign, rsuID
3. Rsu valid or not
4.send vehicle_info
5.verify V_id with V_id^sign
6. V_id valid or not

Figure 3. Phase 1 communication flow

- The V_{id} shares $(\mathscr{P}, \alpha, \beta, \mathscr{H})$ with the RA, while keeping $\mathscr{X}$ private.

This phase of the proposed framework ensures that no malicious vehicle can perform any masquerading attack while sending the information.

4.2. PHASE 1: ENTITY AUTHENTICATION

The event information request is dissipated via the RSUs within the RO. The RSUs dissipate $\mathscr{G}_{\text{public}}$, its digital signature, *rsuSign*, generated using its registered private key, $\text{Prik}_{\text{rsuID}}$ and the *rsuID* within the ROI.

A vehicle, V_{id}, that wants to share its information with *rsuID* communicates with the *RA* using a secured communication channel to determine the authenticity of *rsuID* using its *rsuSign*. If it is found authentic, the vehicles sends the *vehicle_info* packet to the *rsuID* by encrypting it with $\mathscr{G}_{\text{public}}$ as shown in Algorithm 7.

$$vehicle\_info = \mathscr{G}_{\text{public}}(V_{\text{id}}, {V_{\text{id}}}^{\text{sign}})$$

where, V_{id}^{sign} is the digital signature of V_{id} generated as shown below.

$$\mathscr{R} = \alpha^{k}(mod\ \ \mathscr{P}) \tag{3}$$

$$s = k^{-1}(\mathscr{H}(V_{id}||t_{gen}) - \mathscr{X}.\mathscr{R})(mod\ \ \mathscr{P} - 1) \tag{4}$$

$$V_{id}^{sign} = (\mathscr{H}(V_{id}||t_{gen}), \mathscr{R}, s, t_{gen})$$

where k is a secret random integer relatively prime to $\mathscr{P} - 1$ and t_{gen} is the timestamp when the V_{id}^{sign} is generated. The purpose of the t_{gen} is to prevent any malicious RSU from performing replay attack using the V_{id}^{sign} of a non-malicious vehicle later. This has been analyzed in section VI.a. The *vehicle_info* sent by the V_{id}s is stored for future reference.

Algorithm 7 Encryption using $\mathscr{G}_{public}$

```
1: Input: M(vehicle_info/ tokenPkt/ eventPkt), $(G, \mathscr{G}, q, r)$
2: Output: Cipher text, (cipher1,cipher2)
3: Msg: Mapping M$\rightarrow m \in G$, y$\in \{1 \ldots q-1\}$
4: $s = r^{y}$, $cipher1 = \mathscr{G}_{y}$, $cipher2 = Msg \cdot s$
5: return (cipher1,cipher2)
```

The *rsuID* after receiving the *vehicle_info*, decrypts it using $\mathscr{G}_{private}$ as shown in Algorithm 8, authenticates the V_{id} by sending V_{id} and V_{id}^{sign} to the RA. The RA checks if $\beta^{\mathscr{R}}.\mathscr{R}^{s}(mod\ \ \mathscr{P}) \overset{?}{\equiv} (\alpha^{\mathscr{H}(V_{id}\ ||\ t_{gen})}(mod\ \ \mathscr{P}))(mod\ \ \mathscr{P})$. If it matches, V_{id} is declared authentic else it is declared malicious. It is to be noted that the *rsuID* does not know the $(\mathscr{P}, \alpha, \beta, \mathscr{H})$ corresponding to V_{id}. If the V_{id} is authentic, the *rsuID* stores the V_{id}, time of the request and the location of the vehicle as a record for future reference. The communication flow of Phase 1 is shown in Figure 3.

Algorithm 8 Decryption using $\mathscr{G}_{\text{private}}$

```
1: Input: (cipher1,cipher2), $(G, \mathscr{G}, q, r)$, $\mathscr{G}_{\text{private}}$
2: Output: M(vehicle_info/ tokenPkt/ eventPkt)
3: Msg: Mapping M$\rightarrow m \in G$, y$\in \{1...q-1\}$
4: $s = cipher1^{\mathscr{G}_{\text{private}}}$
5: Compute $s^{-1}$ using Extended Euclid's Algorithm [16]
6: Msg= $cipher2 \cdot s^{-1}$
7: M: Map back every $m \in G$ in Msg.
8: return M
```

4.3. PHASE 2: VEHICLES COLLECTING NEIGHBOR PACKETS AND SENDING INFORMATION TO RSU

In this phase, a vehicle generates *tokenPkt*, and dissipates it to the vehicles within its transmission range. Thereafter, the vehicles wait for δ amount of time to receive the *tokenPkt/s* transmitted by the nearby vehicles, which is abbreviated as *tokenPkt'/s*. Subsequently, it generates the *eventPkt* encrypted with $\mathscr{G}_{\text{public}}$ consisting of the information mentioned below and sends it to the RSU. The V_{id} also sends a hash function, *hf*, to the RSUs which is leveraged in *Phase 4*, along with the location of the vehicle, *Loc*. The *event,*V_{id}^{sign}*,timestamp* together is referred to as the information part ($info_{id}$) which is leveraged in *Phase 4.1*. The *timestamp* is the time when *eventPkt* is generated.

$$info_{\text{id}} = \langle event, V_{\text{id}}, V_{\text{id}}{}^{\text{sign}}, timestamp \rangle$$

$$tokenPkt = \mathscr{G}_{\text{public}}(info_{\text{id}})$$

$$eventPkt = \mathscr{G}_{\text{public}}(info_{\text{id}}, L_{\text{id}}, tokenPkt'/s, Loc, \rho, hf, rad)$$

$$\rho = \frac{(n_{\text{vehicles}} * approximate\_ROI\_length)}{vehicle\_sensing\_range}$$

where ρ is the vehicle density calculated by the vehicle, n_{vehicles} is the number of vehicles observed by a vehicle within its sensing range, L_{id} is the lane id of the ROI (1 being the farthest left and n being the farthest right in a n-lane ROI) approximate_ROI_length is the approximate length of the ROI for which the traffic is being monitored, vehicle_sensing_range is the length till which the OBU of the vehicles can sense other vehicles, and *event* is the traffic event (whether congested/non-congested) that is observed by the vehicle. *rad* is the untamperable radar distance (in front of the vehicle) calculated by a $\mathrm{V_{id}}$.

This may happen that a vehicle may perceive more number of vehicles within its transmission range than received *tokenPkt*s. This scenario has been analyzed later in this paper in the *AVD algorithm analysis* under Section VI.(E). Under such a scenario, *rad* is leveraged to identify the number of vehicles within the region.

4.4. PHASE 3: DECISION MAKING BY RSU USING AVD ALGORITHM

The RSUs after receiving the *eventPkt*s from the $\mathrm{V_{ids}}$, decrypts it using Algorithm 8, and check whether the $\mathrm{V_{id}}$ is in the *revoked vehicle list*. If it is present, the RSU discards the information else it accepts it. In *BLAME*, there is a predefined value, $V_{threshold}$, that determines the event of the ROI after receiving the information from the vehicles. In other words, if the total number of vehicles at the ROI detected is greater than $V_{threshold}$, then the ROI is congested else it is non-congested. For determining the number of vehicles at the ROI, we use the *eventPkt* information sent by the vehicles along with the proposed *AVD* algorithm to determine the number of vehicles at the ROI.

The RSUs extracts the event and neighbor information from the *eventPkt* of the $\mathrm{V_{id}}$s using $\mathscr{G}_{\mathrm{private}}$ (Algorithm 8). Thereafter, it forms the *DSG* as defined in [3]. In the *DSG*, any $\mathrm{V_{id}}$, $\mathrm{V_1}$, that has received the *tokenPkt* of another $\mathrm{V_{id}}$, $\mathrm{V_2}$, has an undirected edge between, which shows that they are within the transmission range of each other (neighboring vehicle). For any pair of vehicles, $\mathrm{V_i}$ and $\mathrm{V_j}$, with conflicting *event* recorded, *congested* and *non-congested* respectively, in the *DSG*, we set the *NeA*(V_i,V_j) to *true* (Algorithm 9, lines 4-5) and the AVD of the ROI is calculated using Algorithm 9 (lines 6-11). If the number of $\mathrm{V_{id}}$s returned in Algorithm 9(line 15) is greater than *V_threshold*, then the traffic condition of the ROI is determined as *congested* else it is *non-congested.*

Algorithm 9 NeA and AVD Algorithm

```
1:  Input: eventPkt of all vehicles
2:  Output: NeA if exists, number of $V_{id}$s at ROI
3:  $AVD_0 = \emptyset, k = 1, V_m{}^1$ = neighbors of $V_1$
4:  for number of eventPkt received do
5:      if $V_j \exists V_m{}^i$ then
6:          if $V_i$ and $V_j$ report conflicting events then
7:              $NeA(V_i, V_j) = true$
8:          if $V_i, V_j \notin AVD_{k-1}$ then
9:              $AVD_k = AVD_{k-1} \bigcup (V_i \bigcup V_j)$
10:         else if $V_i \notin AVD_{k-1}$ & $V_j \in AVD_{k-1}$ then
11:             $AVD_k = AVD_{k-1} \bigcup V_m{}^i$
12:         else if $V_j \notin AVD_{k-1}$ & $V_i \in AVD_{k-1}$ then
13:             $AVD_k = AVD_{k-1} \bigcup V_m{}^j$
14:         $K++$
15: return number of $V_{id}$s at ROI
```

In *BLAME*, we use Algorithm 10 to identify the malicious vehicles within the ROI. According to the algorithm, if a vehicle, V_i is in the neighbor list of another vehicle, V_j, then both the vehicles are within the transmission range of each other and recorded conflicting events. Therefore, V_j must also be present in the neighbor list of V_i (Algorithm 10, lines 5-8). If not, V_i is identified as malicious. Algorithm 10 ensures that *BLAME* is resilient to *packet drop attack*.

Additionally, this may happen that the malicious V_{id}s may collude and some of the V_{id}s may appear as ghost vehicle, where a ghost vehicle is defined as a vehicle that is not authenticated with the RSU before entering the ROI and does not broadcast *tokenPkt* within the network. As a result, even if they contribute to a congestion at a particular lane within the ROI, they cannot be traced. Under such a circumstance, a non-malicious V_{id}, V_i, reports higher value of ρ with lesser number of *tokenPkt/s* and report a region as *congested*. To handle this issue where *NeA(V_i,V_j)* is set to *true* for a malicious V_j, the RSU will calculate the average distance between the V_{ids} in a particular lane, $avgRad_i$, using the *rad* information of the V_{id}s as shown in equation 5 and compare it with the safety standards defined in [17], which we term as *safetyDist*. Thus, if the $avgRad$ of a lane is greater than *safetyDist* then the road is *non-congested*, else it is *congested* even if the number of V_{id}s using AVD is determined to be less than $V_{threshold}$, and the V_{id}s reporting contradictory *event* is entered into *revoked_vehicle_list*. The *avgRad* information alone may be sufficient to determine the *event* of the ROI, but insufficient to identify the malicious V_{id}s from the ROI without the AVD algorithm.

$$avgRad = \sum_{i=1}^{n} \sum_{j=1}^{num} \frac{rad_j}{num} \tag{5}$$

where *num* is the number of *tokenPkt/s* or *rad* information received and *n* is the number of lanes in the ROI. In equation 5, we find the distance between the vehicle for each lane.

Algorithm 10 Identifying Malicious Vehicles.

```
1: Input: NeA of any 2 vehicles, $V_i$ and $V_j$
2: Output: revoked_vehicle_list
3: $V_m^l$ = neighbors of $V_l$, revoked vehicle list= $\emptyset$
4: for 2 vehicles, $V_i$ and $V_j$, if NeA($V_i$,$V_j$) do
5:     if $V_i \in V_m^j$ & $V_j \notin V_m^i$ then
6:         revoked vehicle list.add ($V_i$)
7:     else if $V_j \in V_m^i$ & $V_i \notin V_m^j$ then
8:         revoked vehicle list.add ($V_j$)
9: return revoked vehicle list
```

This phase in *BLAME* ensures that the RSUs are able to determine the traffic condition of the ROI using the neighbor information under the influence of malicious vehicles (even under majority presence) or if some malicious vehicles appear as ghost vehicles. Also, this phase ensures that *BLAME* is resilient to packet drop attack, false information injection attack, and masquerading attack.

4.5. PHASE 4: BLOCKCHAIN NETWORK: CONSENSUS MECHANISM, BLOCK GENERATION AND ADDITION, IDENTIFYING MALICIOUS RSUS

In this phase, the RSUs add the block consisting of the $\mathrm{V_{id}}$s that sends the *eventPkt*, the hash of the $info_{id}$ corresponding to the $\mathrm{V_{id}}$, an index to the storage pool, hash of the previous block, and timestamp of when it receives the *eventPkt* from the $\mathrm{V_{id}}$ into the blockchain as shown in Figure 4. For adding a block into the blockchain, every RSUs within the ROI needs to have a consensus about the event, and the malicious vehicles detected. Since the RSUs are connected in the blockchain, the *eventPkt* of the vehicles has to be distributed to all the RSUs so that every RSU can identify the malicious vehicles and determine the traffic scenario of the region independently using Algorithm 9 and 10.

Phase 4.1: POW by the multiple RSUs

If more than one RSU wants to add the same block of information into the blockchain, the RSUs solve a *proof-of-work (POW)*. In our framework, for a *POW*, the RSUs have to generate a hash of the block to be mined based on the *mining difficulty value* and a *nonce* value. A *mining difficulty value* is defined as the hash value generated by the blockchain nodes (RSU) which are preceded by a certain number of *0s*. For example, a mining difficulty value of 4 indicates that the RSUs need to generate a hash of the data block which is preceded by 4 *0s*. A *nonce* value is used while generating the hash and is also used for verification during the consensus algorithm. The RSU that solves the *POW* first, say RSU_{pMiner}, becomes eligible to become the *miner node*, and sends the solution of *POW* to all other RSUs in the blockchain network for consensus.

Phase 4.2: Consensus algorithm among the RSUs

As the RSUs are connected within the blockchain, the information from all the *eventPkt*s along with the RSU that received it is shared among every other RSUs. For providing the consensus, every individual RSU verifies the following:

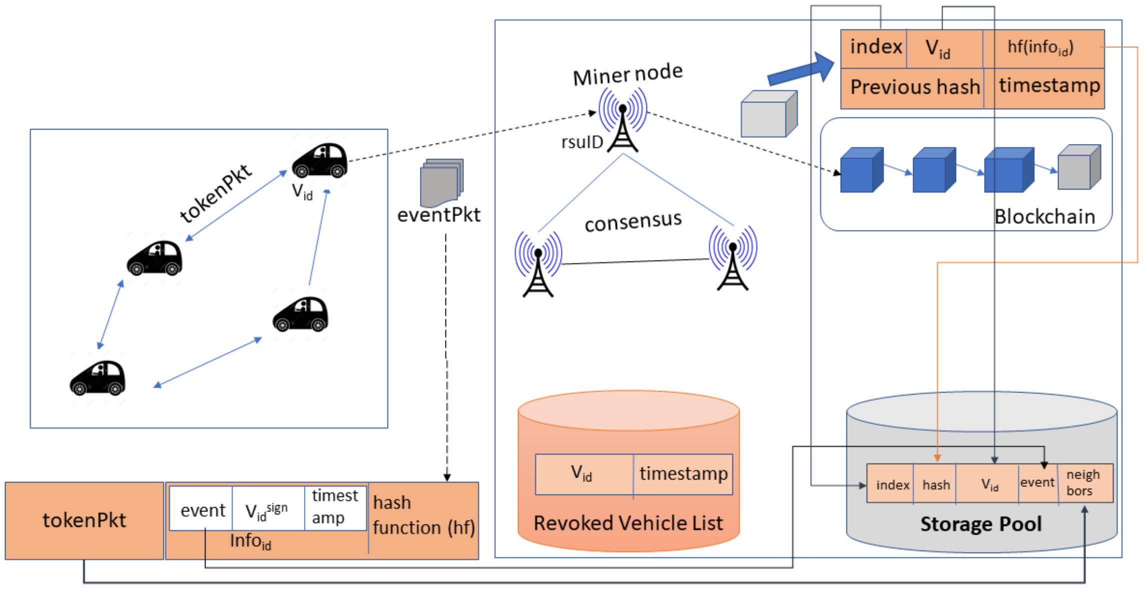


Figure 4. Phase 4 workflow

- If a *complainPkt* (explained in *Phase 4.3*) has been received against RSU_{pMiner} and it is found to be malicious (the possibility of which has been described in the next subsection, Phase 4.3).

- If the POW solution is valid or not, i.e., it has correctly generated the hash using the *nonce* value.

- If the V_{id}^{sign} contained in the *info*$_{id}$ field and *tokenPkt/s* of every *eventPkt* corresponding to a $V_{id} \overset{?}{=}$ valid, through the aid of the RA.

- For every *eventPkt* corresponding to a V_{id}, if *hf(info*$_{id}$*)*, *event* reported by the V_{id} and *timestamp* information matches with the information available to the RSU verifying the block.

- If the *event* detected within the ROI and the *revoked vehicle list* is consistent among all the RSUs.

If all the above criteria satisfy, a consensus from at least the majority of the RSUs within the network is received by RSU_{pMiner}, which adds the data block into the blockchain consisting of the information shown in Figure 4. A corresponding entry with the information of all the V_{id}s is made into the storage pool as shown in Figure 4.

Phase 4.3: Identifying Malicious RSUs

In this phase, a V_{id} downloads the updated blockchain to see whether its information from the *eventPkt* has been registered or not. If it finds that the information has not been registered into the blockchain, it resends the *eventPkt* to the nearby RSU within the ROI. Under such circumstance, the RSUs re-analyze the received *eventPkt*s of the V_{id}s using *AVD* algorithm to determine the *event* of the region. If the re-analyzed *event* is different from the recorded event in the blockchain, *Phase 4.2* and *Phase 4.3* are repeated and a new entry is made into the blockchain.

If it is registered, it checks whether the $hash_{id}$ in the block recorded by RSU_{miner} for the V_{id} matches with its own hash of $info_{id}$. If it matches, then the V_{id} uses the index field of the block to access the storage pool and check whether the event matches with the event sent by the V_{id}. If the event at the corresponding index in the storage pool is the same as recorded by the V_{id}, it takes no further action.

However, if the hash in the block or the event in the storage pool does not match, it sends the *complaintPkt* to the nearby RSU consisting of the contents of the *eventPkt* and the *rsuID* to which it had to send the *eventPkt* in *Phase 2*. The V_{id} will keep updating the latest blockchain and sending the *eventPkt*(or the *complaintPkt* if there is a mismatch) until it is correctly registered into the blockchain.

$$complainPkt = (\langle eventPkt \rangle, complain\_rsuID)$$

where *complain_rsuID* is the *rsuID* to which it had send the *eventPkt* in *Phase 2*.

On receiving the *complaintPkt* from the $\mathrm{V_{id}}$, the RSUs re-analyze the *eventPkt*s of the $\mathrm{V_{id}}$s along with the *complaintPkt* to determine the *event* of the region. If the *event* remains the same as before, it checks whether the information ($hash_{id}$) entered into the blockchain and the event recorded at the corresponding *index* in the storage pool is consistent or not. If it is consistent, the RSUs ignores the *complainPkt* of the $\mathrm{V_{id}}$ and adds $\mathrm{V_{id}}$ into the *revoked vehicle list*. This may happen if the $\mathrm{V_{id}}$ is malicious and maliciously sends *complainPkt* against a *rsuID*. However, if the information is not consistent, the RSU with the *rsuID* mentioned in the *complaintPkt* is marked as compromised and is removed from the blockchain network. Thereafter, *Phase 4.1, 4.2, and 4.3* are repeated, where a new RSU_{pMiner} is selected that updates the blockchain with the new information.

The use of blockchain in *BLAME* brings transparency to the vehicular plane where the decision-making information is available to the $\mathrm{V_{id}}$s unlike the state-of-the-art frameworks, where a vehicle has to solely rely on the decision made by the centralized entity or the RSUs. Furthermore, the timestamped information in the blockchain brings transparency among the RSUs, provides resiliency against repudiation attack (where a malicious RSU may deny adding some incorrect information), and also bring immutability to the *eventpkt* of the $\mathrm{V_{id}}$s. This phase also ensures that the malicious RSUs within the network is not able to perform packet drop attack against any non-malicious $\mathrm{V_{id}}$, and also makes the framework resilient to any false AVD calculation attack that any malicious *rsuID* may perform while determining the *event* of the ROI.

5. EVALUATIONS AND EXPERIMENTAL RESULTS

The effectiveness of the proposed framework has been validated through extensive simulations using Ventos [10] where the road network was simulated using SUMO [18] and the networking was simulated using Omnet++. The blockchain among the RSUs was emulated as shown in [19]. Three machines with the configurations as mentioned in Table 1 was emulated as the RSUs of the network. The efficiency of our proposed framework in identifying the malicious vehicles and determining the traffic condition of the region was tested against different percentages of malicious vehicles within the network using different simulation parameters.

Table 1. Simulation Parameters

Beacon Interval	4 seconds
Number of vehicles	10-50
Communication	Dedicated Short Range Communication (DSRC)
Interference distance of vehicles and RSUs	510.5 metres (default simulator value)
Blockchain node 1(RSU 1) configuration	Intel Core I5-9300H (2.4GHz), 8gb RAM
Blockchain node 2(RSU 2) configuration	Intel Core i3-4005U (1.7GHz), 8gb RAM.
Blockchain node 3(RSU 3) configuration	Intel Xeon E5-1620 (3.7 GHz), 16gb RAM
Bitrate	3 Mbps

Table 2. Security Comparison

Security attacks	[4]	[11]	[7]	[13]	*BLAME*
MITM	✓	✗	✗	✓	✓
Collusion	✗	✓	✗	✗	✓
False data dissemination	—	—	—	—	✓
Non-Repudiation	✓	✓	✓	✓	✓
Majority Malicious Vehicle	✗	✗	✗	✗	✓
On-Off	✓	✗	✓	✓	✓
Ballot Stuffing	✗	✗	✗	✗	✓
Bad Mouthing	✗	✗	✗	✓	✓
Masquerading	✓	✓	✓	✓	✓
Packet Drop	✓	✓	✗	✓	✓

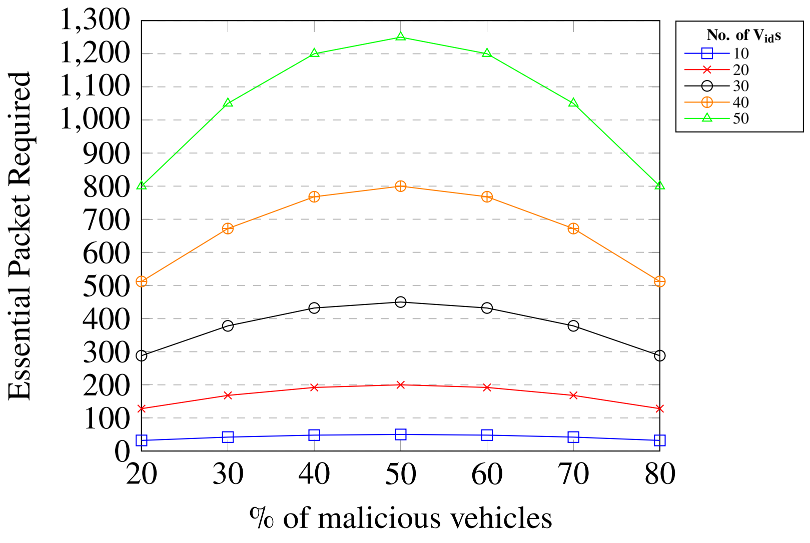


Figure 5. Essential Packet needed for identifying malicious V_{id}

In Table 2, we compare the security features of *BLAME* against some of the event validation state-of-the-art models [4][11][7][13]. As we can see from Table 2, the state-of-the-art models handle some of the attacks listed but not all the attacks at the same time. It is also important to notice that none of the models can provide accurate event information from the ROI if the malicious vehicles form a majority within the network. As a result for all of the proposed models, if the malicious vehicles perform false information dissemination attack within the network, the detection efficacy is highly conditional (represented using —), as they can detect the event only if the majority of the vehicles are non-malicious.

During the simulations, it was seen that the proposed *AVD* algorithm can identify the malicious vehicles and the event from the ROI if the RSUs can receive the *Essential Packets*. *Essential Packets* is defined as the packets that are sent from the malicious vehicles to the non-malicious vehicles and vice versa. To identify the malicious vehicles and the event from the ROI, the number of packets required is determined using equations 6 and 7.

$$Malicious\_vehicles = (x\%\ of\ N) * (y\%\ of\ N) * 2 \tag{6}$$

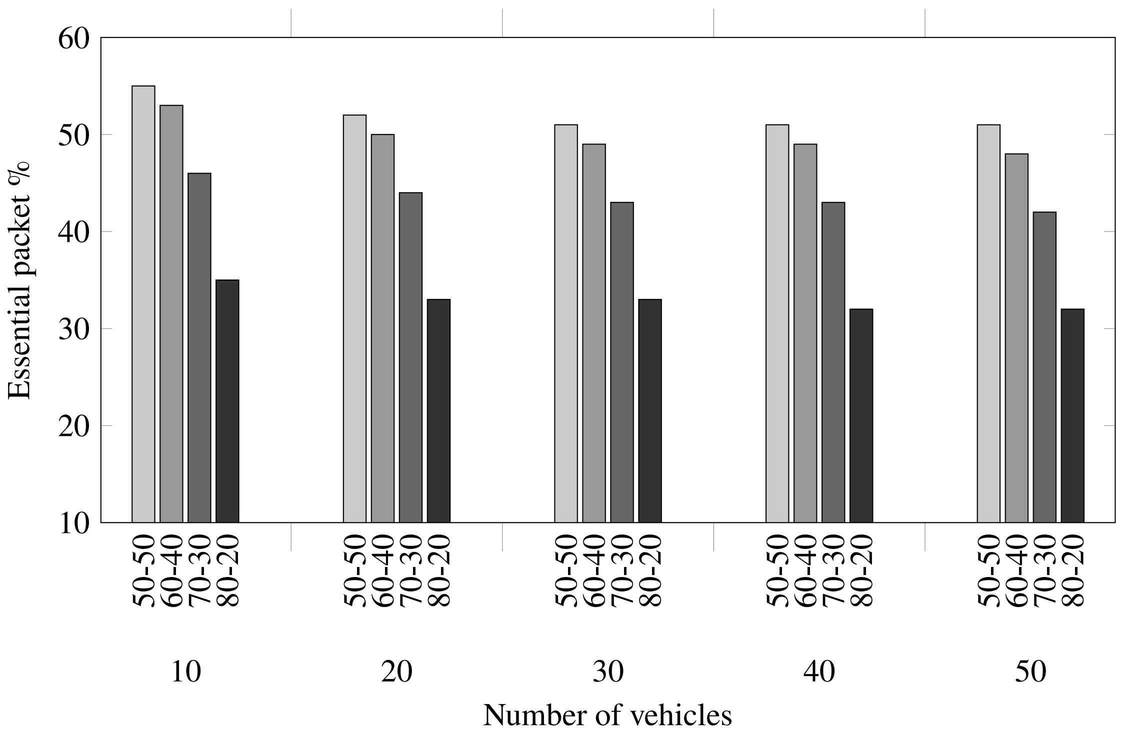


Figure 6. Essential Packet percentage compared to total number of packets under varying malicious vehicles

$$Event = (x\%\ of\ N) * (y\%\ of\ N) \tag{7}$$

In equations 6 and 7, x is the percentage of malicious vehicles while y is the percentage of non-malicious vehicles out of a total number of N vehicles. In other words, to detect the event from the ROI we will need to obtain the packets only broadcast by the non-malicious vehicles. This may result in incorrect identification of malicious vehicles. Thus for accurate identification of the malicious vehicles, we need to obtain all the *Essential Packets*. It can be seen from Figure 5 that the requirement of *Essential Packets* for correct identification of the malicious vehicles within the network increases as the number of vehicles increases. This is because there is more number of packets exchanged between the malicious and the non-malicious vehicles. It can also be seen that with the increase in the number of vehicles, as the difference between the percentages of malicious and non-malicious vehicles in the network decreases, it results in a bell-shaped curve. This is because the lesser difference in percentages signifies approximately an equal number of malicious and non-malicious vehicles within the network. As per the definition of the *Essential Packet*, this results in a higher generation of essential packets within the network, which is required to identify the malicious vehicles. However, it can also be seen from Figure 6 that the scalability of the proposed framework concerning the size of the network remains uniform. It can be seen from Figure 6 that even if the number of vehicles within the network increases, the identification of the malicious vehicles and the event within the network requires almost the same percentage of packets (*Essential Packets*) compared to the total number of packets within the network. Thus, the proposed model is suitable for urban to the rural transportation system. The number of *Essential Packets* required to detect the event is half than that is required to identify the malicious V_{id}s. Hence, if the malicious V_{id}s are identified, the event from the ROI is detected accurately.

Table 3. Vehicle Detection Accuracy

Attack Type	Mean Accuracy (in %)
Type-I	100
Type-II	78.5
Type-III	49.7

Table 4. Essential Packet Receiving Rate

Parameters	Essential Packet Received	Parameters	Essential Packet Received
$\langle 5,5,10 \rangle$	Yes (100%)	$\langle 15,5,20 \rangle$	No (65.7%)
$\langle 10,5,10 \rangle$	No (77.8%)	$\langle 15,10,20 \rangle$	No (78.6%)
$\langle 10,10,10 \rangle$	No (93.3%)	$\langle 15,15,20 \rangle$	No (88.1%)
$\langle 10,15,10 \rangle$	Yes (100%)	$\langle 15,20,20 \rangle$	No (94.8%)
$\langle 15,5,10 \rangle$	No (65.24%)	$\langle 15,25,20 \rangle$	COR
$\langle 15,10,10 \rangle$	No (73.3%)	$\langle 15,20,30 \rangle$	No (94.8%)
$\langle 15,15,10 \rangle$	COR	$\langle 15,25,30 \rangle$	Yes (100 %)

The performance of our AVD algorithm in detecting the number of vehicles at the ROI was also tested using the *NGSIM* dataset (U.S. Department of Transportation Federal Highway Administration 2016) available in [20]. Since, it is not an attacked dataset, we simulated every odd numbered V_{id} as an attacker and calculated the number of vehicles at the ROI using AVD by leveraging the *Global Time* information available under the influence of the malicious V_{id}s. The type of attacks simulated are described below:

- Type I attack: In this attack, all the malicious V_{id}s at every *Global Time* are modified to drop the *tokenPkt* of the *Preceding and Following* V_{id} within the dataset, where the *tokenPkt* comprises of the V_{id} contained in the *Preceding and Following* field. However, they send the *eventPkt* to the RSU.

- Type II attack: In this attack, all the malicious V_{id} at every *Global Time* does not send the *eventPkt* to the RSU. However, they send the *tokenPkt* to the neighboring V_{id}. This is simulated by dropping the odd-numbered V_{id} from the dataset.

- Type III attack: In this attack, the malicious V_{id} does not send the *tokenPkt* as well as *eventPkt* to the RSU, and they appear as ghost vehicle. Thus, the *Essential Packets* are dropped within the network. This is simulated by dropping the odd-numbered V_{id} from the dataset as well as removing the odd-numbered V_{id} from the *Preceding and Following* field of any other V_{id}.

As we can see from Table 3, the accuracy of identifying the malicious vehicles from the ROI decreases from Type I attack to Type III attack. In Type, I attack, even if the malicious V_{id}s drop the *Preceding and Following* V_{id}s, they send the information to the RSU. As a result, using the AVD algorithm, the RSUs are capable of detecting the total number of vehicles at each *Global Time* using the *Preceding and Following* V_{id} information reported by the non-malicious V_{id}s, and the number of V_{id}s that sent the information to the RSU. Thus, the accuracy of identifying the malicious V_{id}s is the highest (100%). In case of a Type II attack, the malicious V_{id}s does not send the *eventPkt* to the RSU. Assuming that they did not authenticate with the RSU as described in *Phase 1*, they appear as ghost vehicles. However, as they share the *Essential Packets* with the non-malicious V_{id}s, they are detected using the AVD algorithm. There are some malicious V_{id} that appears only as the *Preceding and Following* V_{id} of another malicious V_{id}. As a result, those vehicles become ghost vehicles and are not identified using the AVD algorithm, thus, decreasing the accuracy (78.5%) of identifying the malicious vehicles within the ROI. In Type III attack, the *Essential Packets* from the malicious V_{id} to the non-malicious V_{id} is not received. As a result, only the non-malicious V_{id}s are identified within the ROI using the AVD algorithm. As a result, the accuracy of identifying the malicious vehicle is the least (49.7%) in Type III attack. Nonetheless, the accuracies in Table 3 represent the worst-case accuracy for each attack type. This means that in Type II and Type III attack, if some of the *Essential Packets* and *eventPkts* are detected by the non-malicious V_{id} and the RSUs respectively, that will increase the accuracy of identifying the malicious V_{id}s within the ROI.

In Table 4, we analyzed the percentage of *Essential Packets* received under different parameters represented as $\langle VCount, Bcount, tElapsed \rangle$ where VCount is the total number of vehicles at the ROI, BCount is the number of times *tokenPkt* is broadcast by every V_{id} and tElapsed is the time that a V_{id} waits before sending the *eventPkt* to the RSU. *Essential Packet received* refers to the percentage of *Essential Packets* obtained by the V_{id}s. It can be seen that when VCount is 5, the *Essential Packet* received percentage is high (100%) with lesser values of Bcount and tElapsed. This is because given the transmission range of a V_{id}, it does not take much time for a V_{id} to receive the *tokenPkt* of another V_{id}. However, as the number of V_{id}s increase, the value of BCount and tElapsed needs to be increased to achieve *100% Essential Packet* received rate. We prioritize higher BCount over tElapsed to keep the solution within real-time. However, as the network size increases (i.e., VCount increases), more packets are lost due to network congestion (Packet drop). With an increase in packet drop, in many scenarios, the *eventPkt* does not reach the RSU for decision making (represented by COR in Table 4), and hence, the accuracy of malicious vehicle identification

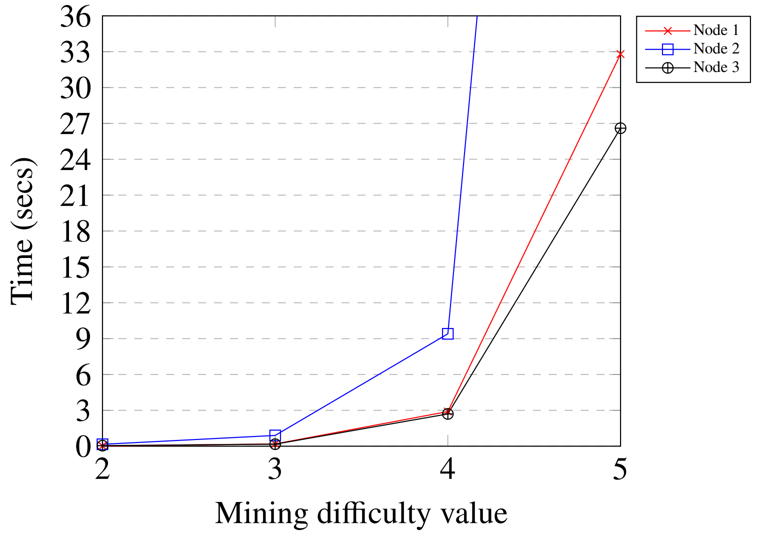


Figure 7. CPU time consumption for mining a block of data

drops or some vehicles non-maliciously become ghost vehicles. Thus, from Table 4, it can be inferred that to keep the solution in real-time, if the network size increases BCount needs to be increased, which will lead to higher packet drop due to network congestion. However, if the BCount is lesser, the tElapsed needs to be increased so that a V_{id} can broadcast *tokenPkt* with a higher delay which can lead to lesser packet collision (i.e., lesser packet drop). This limitation can be handled by dividing the ROI with a large network into several micro ROIs, then applying the proposed solution to find the *event* at each of the micro ROI and combining each result to determine the *event* of the entire ROI. Alternatively, instead of DSRC, can use 5GLTE with a wider range and bandwidth.

We also analyzed the time taken to mine the block into the simulated blockchain network after solving the *POW* using the *mining difficulty* value. We determined the time taken with different mining difficulty values in 3 different simulated nodes. It can be seen from Figure 7, that the time taken for mining increases as the mining difficulty value is increased beyond 4. So, to keep the mining, hence the overall working of the framework close to real-time, we kept the mining difficulty to 4 for the simulations.

6. BLAME ANALYSIS

6.1. FALSE INFORMATION INJECTION ATTACK

Let the correct event from the ROI be *correctEvent*. For the malicious vehicle to perform the false information injection attack, it has to manipulate the *event* field, *falseEvent*, of the $info_{id}$, contradicting *correctEvent* to make it appear legitimate. It can either: 1) Add *tokenPkt/s* into the *eventPkt* it generates, or 2) Discard some legitimate *tokenPkt/s* within the network from including into the *eventPkt* it generates.

6.1.1. Add *tokenPkt/s* into the *eventPkt*. In this scenario, the attacker wants to show that the ROI is congested when it is non-congested. For this attack, it has to generate more random *tokenPkt/s*, *token_random*.

Let us assume the attacker generates random *tokenPkt* with $\mathrm{V_{id}}^{\mathrm{sign}}$, say $\mathrm{V_{random}}^{\mathrm{sign}}$, corresponding to some legitimate $\mathrm{V_{id}}$, say $\mathrm{V_{random}}$. When RSU receives the *eventPkt* from the attacker, it sends the $\mathrm{V_{random}}^{\mathrm{sign}}$ and $\mathrm{V_{random}}$ to the RA. The RA finds that $\beta^{\mathscr{R}}.\mathscr{R}^{\mathrm{s}}(mod\ \mathscr{P}) \not\equiv (\alpha^{\mathscr{H}(\mathrm{V_{id}} \,||\, \mathrm{t_{gen}})}(mod\ \mathscr{P}))(mod\ \mathscr{P})$ still holds as $\mathscr{P}$, β, $\mathscr{H}$ and α remains unknown to the attacker. In case of a naive attack approach, even if $\mathrm{V_{random}}$ is not legtimate/ randomly generated by the attacker, the RA finds that $\mathrm{V_{random}}$ is not registered and discards the packet.

Thus the proposed framework can detect an attack where the attacker (vehicles or RSU) adds random *tokenPkt/s* to manipulate the *event* of the ROI.

6.1.2. Discard Legitimate *tokenPkt/s* from Non-Malicious Vehicles. In this scenario, the attacker wants to show that the ROI is non-congested when it is congested.

Let us assume the number of malicious vehicles be *mCount* while the number of non-malicious vehicle be *nmCount*. The malicious vehicles reports *mCount tokenPkt/s* within its *eventPkt* where m*mCount* is less than *threshold* that is used to determine congestion. The $\mathrm{V_i}$ of the *tokenPkt/s* recorded by the malicious vehicle, $\mathrm{V_{mal}}$ is added into AVD list if it does not exist and $\mathrm{V_i}$ is valid. For any non-malicious vehicle, $\mathrm{V_{nonMal}}$ that does not drop any *tokenPkt*, if $\exists$ $V_k \in$ *tokenPkt* of V_{nonMal} where V_k=valid, then we set NeA(V_k,V_{nonMal}) to be *true*, as they report conflicting event and add them into AVD list. Furthermore, as V_k drops all non malicious *tokenPkt/s*, $\exists$ $V_{nonMal} \notin$ *tokenPkt* of V_k even though V_{nonMal}=valid. Thus, V_k is identified as malicious.

Hence, the proposed framework can detect the number of vehicles as AVD eventually consists of all the V_{id}s that create a token packet and identify the malicious vehicles.

6.2. MASQUERADING ATTACK

For a malicious vehicle to perform masquerading attack on a target vehicle, it has to regenerate or acquire $\mathscr{P}$, $\mathscr{H}$ and α that was shared by the target vehicle with the RA during the registration phase using a secured channel, as well as $\mathscr{X}$ and k that is kept secret by the target vehicle. As we can see from eqn 2, the difficulty of finding $\mathscr{X}$ from β is based a discrete logarithm problem for a large prime number $\mathscr{P}$, for which there exists no efficient solution. Furthermore, in *Phase 1*, if the attacker sends an arbitrary value of $\mathscr{R}$ and s inside V_{id}^{sign}, $\beta^{\mathscr{R}}.\mathscr{R}^{s}(mod \;\; \mathscr{P}) \not\equiv (\alpha^{\mathscr{H}(V_{id} \, || \, t_{gen})}(mod \;\; \mathscr{P}))(mod \;\; \mathscr{P})$ still holds as $\mathscr{P}$, β, $\mathscr{H}$ and α remains unknown to the attacker. Thus the proposed framework is resilient to masquerading attack.

6.3. MAN-IN-THE-MIDDLE (MITM) ATTACK

By performing the MITM attack, an attacker (or a malicious vehicle) tries to breach the confidentiality and integrity of the *vehicle_info*, *tokenPkt*, and *eventPkt* that is broadcasted by other nearby vehicles. Given the value of r in $\mathscr{G}_{public}$ which is available as the public key, to reveal the content of the packets mentioned above, the attacker has to obtain $\mathscr{G}_{private}$, which is stored only among the RSUs, the difficulty of which is based on discrete logarithm problem.

Thus, the proposed framework is resilient to MITM attacks and also prevents any breach of integrity and confidentiality of a vehicle's data from the nearby vehicles.

6.4. REPLAY, PACKET DROP, OR FALSE INFORMATION BLOCK ATTACK

To perform replay attack, the malicious RSU has to store the $\mathrm{V_{id}}^{\mathrm{sign}}$ of any legitimate $\mathrm{V_{id}}$ for validation by the RA. However, if the malicious RSU stores $\mathrm{V_{id}}^{\mathrm{sign}}$ for a legitimate $\mathrm{V_{id}}$ at $t_{\mathrm{gen}} = t_1$, the RA can still find $\beta^{\mathscr{R}}.\mathscr{R}^{\mathrm{s}}(mod\ \ \mathscr{P}) \not\equiv (\alpha^{\mathscr{H}(\mathrm{V_{id}}\,||\,\mathrm{t_{gen}})}(mod\ \ \mathscr{P}))(mod\ \ \mathscr{P})$ as $\mathscr{P}$, β, $\mathscr{H}$ and α remains unknown to the malicious RSU. As $\mathscr{H}$ is a function of $\mathrm{V_{id}}$ and t_{gen}, the malicious RSU cannot generate a valid $\mathscr{H}(V_{\mathrm{id}}||t_{\mathrm{gen}})$ when $t_{\mathrm{gen}} = t_2$ without knowing $\mathscr{H}$, known only by the RA and the legitimate $\mathrm{V_{id}}$.

To add false information within the blockchain, the RSU can either drop an *eventPkt* corresponding to a non-malicious $\mathrm{V_{id}}$, add incorrect *event* corresponding to a legitimate $\mathrm{V_{id}}$ within the storage pool, add incorrect *hf(info$_{id}$)* corresponding to a legitimate $\mathrm{V_{id}}$. In our proposed framework, *Phase 4.2*, a $\mathrm{V_{id}}$ downloads the updated blockchain to verify if its information from *eventPkt* has been recorded. If not, it resends the *eventPkt* to the nearby RSU. As the majority of the RSUs within the ROI is non-malicious, the moving $\mathrm{V_{id}}$ eventually sends *eventPkt* to a non-malicious RSU which is updated into the blockchain. If the malicious RSU alters some information of the *info$_{id}$* field, *hf(info$_{id}$)* generated by the non-malicious $\mathrm{V_{id}}$ ≠ *hf(info$_{id}$)* generated by the malicious RSU for non-malicious $\mathrm{V_{id}}$. Under such circumstance, the non-malicious $\mathrm{V_{id}}$ generates a *complainPkt* and sends to the nearby RSU where the non-malicious RSUs recalculate the *hash$_{id}$* of the $\mathrm{V_{id}}$ along with the *event* using AVD and remove the malicious RSUs from the network. Even if the malicious RSU does not alter *hash$_{id}$* of the non-malicious $\mathrm{V_{id}}$ but changes the *event* recorded by $\mathrm{V_{id}}$ in the storage pool, the $\mathrm{V_{id}}$ can detect it using the *index* field in the updated blockchain and referring to the corresponding index in the storage pool. This situation is likely in scenarios where the distribution of RSUs are not uniform (or sparse, such as in rural or semi-urban transportation system) and no non-malicious RSU falls within the range of transmission of a non-malicious vehicle. Thus, the proposed framework can detect replay attacks, packet drop attacks, and false information added to the blockchain by the malicious RSU.

6.5. COLLUSION ATTACK

To perform the collusion attack, the attacker may do the following: Case-1) Drop the *tokenPkt* of all the non-malicious vehicles. Case-2) a Selective number of malicious V_{id}s may appear as ghost vehicle, i.e., do not authenticate with any RSUs at the ROI in *Phase 1*.

For Case-1, we have analyzed in section VI a. that even if the malicious V_{id}s drop the *tokenPkt/s* of the non-malicious V_{id}s, the proposed AVD algorithm is effective.

For Case-2, the remaining malicious V_{id}s may send the *event* as 'non-congested' when the actual *event* is 'congested' or vice-versa. However, the RSU calculates the distance between the vehicles using the *rad* information of all the V_{id}s in a lane and compares it with the $avgRad$ (equation 5) to determine the *event* of the region. Even if the ghost V_{id}s do not send the *tokenPkt/s*, the *rad* information of the vehicle provides evidence of an object in-front of it, even if there is no record of a expected vehicle in the *Loc*. Thus, the proposed framework handles collusion attacks and detects the *event* of the region under the influence of ghost vehicles.

7. CONCLUSION

We have proposed a distributed event validation framework named *BLAME*, that is capable of identifying the true traffic event at the ROI under the influence of malicious vehicles (even if they form a majority or appear as a ghost vehicle). In *BLAME*, we have proposed *AVD* algorithm that is capable of finding the total number of vehicles at the ROI even if some of the malicious vehicles do not send their information to the RSUs or other nearby vehicles, and also leverage the radar information recorded by the vehicles to determine the event at each lane of the ROI. To bring transparency to the vehicular plane, we deployed blockchain among the RSUs (the majority of which are non-malicious) where information of the vehicles are added as a block by the miner node (RSU) after receiving consensus from the other nodes (RSUs). The updated blockchain information is accessible to a vehicle and consequently, a malicious RSU is removed from the network during the consensus algorithm if it exists. The proposed model can handle packet drop attacks, collusion attacks, MITM attacks, and masquerading attacks.

REFERENCES

[1] A. Greenberg, "Hackers remotely kill a jeep on the highway—with me in it," *Wired*, vol. 7, p. 21, 2015.

[2] D. Manivannan, S. S. Moni, and S. Zeadally, "Secure authentication and privacy-preserving techniques in vehicular ad-hoc networks (vanets)," *Vehicular Communications*, 2020.

[3] A. Roy and S. Madria, "Secured traffic monitoring in vanet," *arXiv preprint arXiv:1909.10057*, 2019.

[4] P. K. Singh, R. Singh, S. K. Nandi, K. Z. Ghafoor, D. B. Rawat, and S. Nandi, "Blockchain-based adaptive trust management in internet of vehicles using smart contract," *IEEE Transactions on Intelligent Transportation Systems*, 2020.

[5] N. Malik, P. Nanda, A. Arora, X. He, and D. Puthal, "Blockchain based secured identity authentication and expeditious revocation framework for vehicular networks," in *17th IEEE International Conference On Trust, Security And Privacy In Computing And Communications*, pp. 674–679, IEEE, 2018.

[6] C. Pu, "Blockchain-based trust management using multi-criteria decision-making model for vanets," 2020.

[7] A. Mostafa, "Vanet blockchain: A general framework for detecting malicious vehicles," *J. Commun*, vol. 14, no. 5, pp. 356–362, 2019.

[8] H. Khelifi, S. Luo, B. Nour, H. Moungla, S. H. Ahmed, and M. Guizani, "A blockchain-based architecture for secure vehicular named data networks," *Computers & Electrical Engineering*, vol. 86, p. 106715, 2020.

[9] N. Malik, P. Nanda, X. He, and R. P. Liu, "Vehicular networks with security and trust management solutions: proposed secured message exchange via blockchain technology," *Wireless Networks*, pp. 1–20, 2020.

[10] M. Amoozadeh, H. Deng, C.-N. Chuah, and H. M. e. a. Zhang, "Platoon management with cooperative adaptive cruise control enabled by vanet," *Vehicular communications*, vol. 2, no. 2, pp. 110–123, 2015.

[11] F. Ayaz, Z. Sheng, D. Tian, G. Y. Liang, and V. Leung, "A voting blockchain based message dissemination in vehicular ad-hoc networks (vanets)," in *IEEE International Conference on Communications (ICC)*, pp. 1–6, IEEE, 2020.

[12] Z. Lu, Q. Wang, G. Qu, and Z. Liu, "Bars: a blockchain-based anonymous reputation system for trust management in vanets," in *17th IEEE International Conference On Trust, Security And Privacy In Computing And Communications/12th IEEE International Conference On Big Data Science And Engineering (TrustCom/BigDataSE)*, pp. 98–103, IEEE, 2018.

[13] Z. Lu, W. Liu, Q. Wang, G. Qu, and Z. Liu, "A privacy-preserving trust model based on blockchain for vanets," *IEEE Access*, vol. 6, pp. 45655–45664, 2018.

[14] U. Javaid, M. N. Aman, and B. Sikdar, "Drivman: Driving trust management and data sharing in vanets with blockchain and smart contracts," in *IEEE 89th Vehicular Technology Conference (VTC2019-Spring)*, pp. 1–5, IEEE, 2019.

[15] T. ElGamal, "A public key cryptosystem and a signature scheme based on discrete logarithms," *IEEE transactions on information theory*, vol. 31, no. 4, pp. 469–472, 1985.

[16] S. Parthasarathy, "Multiplicative inverse in mod (m)," *Algologic Technical Report*, no. 1, pp. 1–3, 2012.

[17] FMCSA, "CMV Driving Tips - Following Too Closely." `https://www.fmcsa.dot.gov/safety/driver-safety/cmv-driving-tips-following-too-closely#`, 2015. [Online; accessed 11-February-2015].

[18] P. A. Lopez, M. Behrisch, L. Bieker-Walz, J. Erdmann, Y.-P. Flötteröd, R. Hilbrich, L. Lücken, J. Rummel, P. Wagner, and E. WieBner, "Microscopic traffic simulation using sumo," in *21st International Conf. on Intelligent Transportation Systems (ITSC)*, pp. 2575–2582, IEEE, 2018.

[19] D. Nash, "BUILD A BLOCKCHAIN WITH C++." `https://github.com/teaandcode/TestChain`, 2017. [Online; accessed 20-October-2017].

[20] USDOT, "Next Generation Simulaation dataset." `http://doi.org/10.21949/1504477`, 2016. [Online; 2016].

IV. CANSAFE: AN MTD BASED APPROACH FOR PROVIDING RESILIENCY AGAINST DOS ATTACK WITHIN IN-VEHICLE NETWORKS

Ayan Roy and Sanjay Madria
Department of Computer Science
Missouri University of Science and Technology
Rolla, Missouri 65401
Email: ar3g3@mst.edu and madrias@mst.edu

ABSTRACT

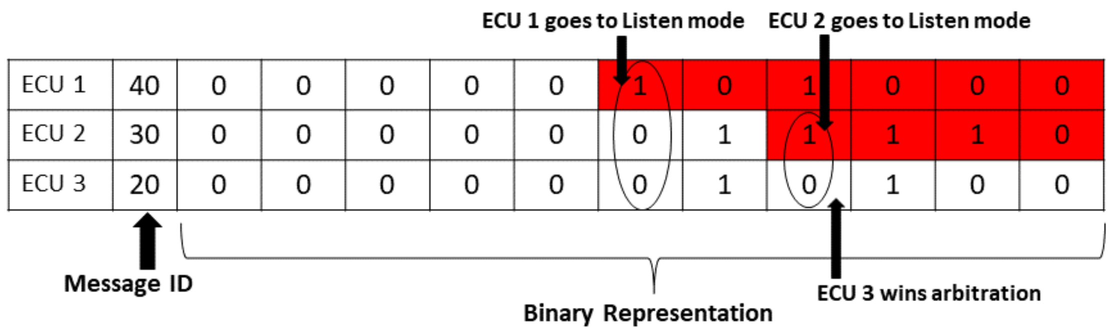
Trending towards autonomous transportation systems, modern vehicles are equipped with hundreds of sensors and actuators that increase the intelligence of the vehicles with a higher level of autonomy, as well as facilitate increased communication with entities outside the in-vehicle network (for example other vehicles or passengers), such as through an infotainment system. However, increase in a contact point with the outside world has exposed the controller area network (CAN) of a vehicle to remote security vulnerabilities. In particular, an attacker can inject fake high priority messages within the CAN through the contact points, while preventing legitimate messages from controlling the CAN (Denial-of-Service (DoS) attack). In this paper, we propose a Moving Target Defense (MTD) based mechanism to provide resiliency against DoS attack, where we shuffle the message priorities at different communication cycles, opposed to the state-of-the-art message priority setup, to nullify the attacker's knowledge of message priorities for a given time. The performance and efficacy of the proposed shuffling algorithm has been analyzed through implementation under different configuration, and compared against the stat-of-the-art solutions. It is observed that the proposed mechanism is successful in denying DoS attack when the attacker is able to bypass preemptive strategies and inject messages within the in-vehicle network.

	Start of Frame (SOF)	Message/ Arbitration ID	Control Field	Message	CRC	ACK	End of Frame (EOF)
Bits:	**1**	**11-29**	**6**	**0-64**	**16**	**2**	**7**

Figure 1. General Structure of CAN message.

1. INTRODUCTION

Modern-day vehicles are equipped with hundreds of electronic control units (ECUs) that manage different functionality of the vehicles, such as the brakes, steering, and throttle. The ECUs within a vehicle form a network called the in-vehicle network (IVN), where they share different kinds of information that determines the state of the vehicle or the next action of the vehicle for a given state. Multiple IVNs are known for in-vehicle communications, such as control area networks (CAN), such as controller area network (CAN), local interconnected network (LIN), and FlexRay. However, CAN is widely accepted as a defacto standard for inter-vehicle communication due to its stability and cost-effectiveness [1]. CAN is a serial communication system where multiple ECUs connected to it share the same physical communication bus. As a result, to avoid message collision, CAN specify an inverse relationship between the message IDs (a CAN message structure is shown in Figure 1) and the message priorities. Thus, at any given time if more than one ECU participates in the message arbitration, the ECU with the lowest message ID wins the arbitration and sends the message through the CAN (as illustrated in Figure 2). Meanwhile, the other ECUs switch to listening mode and wait for the next arbitration, thus providing a non-destructive way of message communication. Furthermore, the modern vehicles are connected to various other entities outside the in-vehicle environment using Bluetooth and 3G/4G networks, such as through the infotainment system [2]. Such remote communication with the outside world aims to enhance the driver's comfort and safety, as well as facilitates in keeping the various software components of the vehicle up-to-date. regular over-the-air (OTA) updates.



ECU 1	40	0	0	0	0	0	1	0	1	0	0	0
ECU 2	30	0	0	0	0	0	0	1	1	1	1	0
ECU 3	20	0	0	0	0	0	0	1	0	1	0	0

Figure 2. Message Arbitration mechanism between message IDs 40,20 and 30. Red box indicates that the ECU goes to listening mode as it has a higher ID than the other messages in the arbitration.

However, CAN is highly vulnerable to remote security attacks, as it does not have any authentication mechanism in place. As a result, it is not difficult for an attacker to connect an external device and snoop into the data packets of the CAN. Furthermore, it can reverse engineer the information collected [3] (more commonly called reconnaissance) and inject information within the CAN. The practicability of remote injection inside CAN has been demonstrated in [4], in which the authors were able to control the braking and steering mechanism of a vehicle through remote injection of data packets. A remote attacker can exploit the strict relationship specification between the message ID and the priority in CAN and effectively perform a DoS attack by always injecting a message with the lowest ID at every time interval, thereby gaining control of the CAN at every time. This can deny legitimate ECUs from sending important information to other ECUs, thus, posing threat to the safety of the passengers and its surrounding.

Moving Target Defense (MTD) is a potential security solution to many static systems [5], in which the system (or attack surface) parameters are constantly changed to nullify the information gathered during the reconnaissance phase of an attacker. The primary steps for MTD based techniques involve: What to dynamically change in the system configurations, When to make the change into the new state from the current state, and How to change the moving attributes to increase the unpredictability of the attack surface. MTD-based security techniques involve using an existing mechanism that is triggered either during certain time intervals (time-based) or when certain events occur (event-based). Many prominent MTD-based techniques involve game theory, network topology (or IP address) shuffling, and machine learning-based based solutions [6].

In this paper, we propose an MTD-based solution to prevent the DoS attack within the in-vehicle network. We reshuffle the relationship between the message/arbitration ID generated in a separate module, referred to as the Shuffling Module, and their priorities at every instant of time to confuse the attacker about the message priorities (as opposed to the specification of the CAN bus). As the specifications of the CAN bus are static, a separate security module/hardware (refer to as SMOD throughout the paper) is needed to handles the message arbitration phase when one or more ECUs wants to send information through the CAN. The detailed specification of SMOD is outside the scope of this paper. However, once completed, the message is sent from the SMOD to the CAN bus. The primary contribution of the paper is as follows:

- We propose a new technique that does not necessarily follow a strict relationship between the arbitration ID and their priorities as described in the CAN specifications. The proposed shuffling algorithm is executed at the Shuffling Module that generates a new arbitration ID for every requesting ECU at a given communication cycle while keeping the relationship between the priorities of the ECUs intact as the original CAN specification.

- We provide a detailed analysis of the proposed shuffling algorithm under different threat models from an attacker. Furthermore, we also compare the various scenarios of the proposed mechanism under different arrangement of the parameters (such as how often to perform the shuffling operation) to determine the impact of the shuffling on the overall in-vehicle network.

2. RELATED WORK

Over the past few years, a significant amount of research has been devoted to the security of the CAN in general. Most the research is based on designing a prem-emptive solution.

In Pre-emptive solutions, the main intention is to prevent any attack into the CAN of a vehicle. Machine/Deep Learning based solutions have been widely explored as potential pre-emptive solutions by many researchers. For instance, the authors in [7] have analyzed the detection performance of Pearson correlation, k-means clustering and Hidden Markov Model in an in-motion vehicle with injected speed and revolutions per minute (RPM) readings. The authors concluded that the use of fabricated (or simulated) dataset incorrectly indicates the performance of an intrusion detection technique. The authors in [8][9] [10][11][12] have proposed different supervised/unsupervised machine learning solutions for detecting intrusion as a pre-emptive measure within the CAN.

Beside the machine learning models, authors have also analyzed the feasibility of graph based solutions [13][14] for detection injection attacks. The authors in [13] have proposed a message sequence based solution to detect an abnormality in pattern between the messages exchanged between the ECUs and determine the injection attack in vehicle. However, the proposed solution relies heavily on an assumption that the normal driving behavior remains constant and any abnormal driving behavior (which may not be injected but different driving condition) is an attacked state. Furthermore, the solution does not take into account that the pattern of messages sent within the in-vehicle is a public information, and can be exploited by a sophisticated attacker. Many researchers [15][16][17] have leveraged the frequency of arrival of messages in CAN as a potential pre-emptive solution for message injection or DoS. In the proposed solutions, the authors assume that every message within the CAN arrive after periodic intervals, that is unknown to the attacker. Thus, the message injected by an attacker adds to an outlier to the periodic interval, which helps in detecting an injected message. However, it is to be noted that the behavior of the

ECUs (or the messages originating from the ECUs) is quite inconsistent with respect to their arrival time, on the advent of which, the efficacy of the proposed models reduces. Furthermore, the efficacy of the model reduces if the attacker is able to mimic the original message frequency. Researchers have also exploited the idea of information theory and entropy-based mechanism for detecting message injection within the network. Very similar to the concept of periodicity, the authors [18][19] here assume that the regular CAN messages have a stable entropy, and a deviation from normal behavior is categorized as an attack by the authors.

An IP-shuffling based MTD defense mechanism has been introduced by the authors in [20] where the authors dynamically changes the default IP address of the ECUs to introduce uncertainty or nullify any reconnaissance of an attacker. However, it is to be noted that the proposed approach serves the purpose of preserving the confidentiality of the ECU from the attacker but does not stop the attacker from performing a DoS (noted by authors also). The authors in [21] has proposed CIST to leverage the use of a fixed priority ID and a dynamic ID to nullify the reconnaissance of an attacker, and provide resiliency against replay and impersonation attack. However, the proposed mechanism does not consider DoS attack as part of their threat model, which breaks the efficacy of the system if the attacker knows the fixed priority ID and replays it at different communication cycles. The authors in [22] have constructed a deterministic finite automation (DFA) for the bit-by-bit arbitration process where a firewall deployed within the system rejects any fake ID injected by an attacker if it is not contained within the whitelist of the ECU IDs maintained inside the firewall. However, the efficacy of the proposed DFA fails if an attacker gathers information through CAN sniffing to reverse engineer and gather some ECU IDs from the whitelist as the IDs remain fixed. Such a captured ID when replayed in separate communication cycle, will be considered as an authentic state by the proposed DFA.

To summarize, it can be observed that most of the preemption mechanism can be used effectively to deny message injection within the network. However, many solutions proposed are more content-based, meaning that it relies on the content of the message to detect if the message is an injected message or not. Such solutions are considered reactive mechanism, which may serve the purpose of detecting an injection attack. However, such solutions does not guarantee resiliency against DoS attack. This is because for injecting a message, as per the CAN specification, the attacker has to win the arbitration among the contending ECUs in a communication cycle. Thus, if the attacker is able to inject the payload, it means that it has successfully won the arbitration and prevented other authenticate ECUs from sending information during that communication cycle. Different from the existing mechanism, we propose a preemptive MTD based solution for the arbitration mechanism where the attacker will not be able to win the arbitration during a communication based on information gathered in previous communication cycles or brute-force approaches.

3. PRELIMINARIES, THREAT MODEL AND ASSUMPTIONS

3.1. PRELIMINARIES

- Shuffling Module : The purpose of the Shuffling Module in the proposed mechanism is to randomly generate the IDs corresponding to different ECUs at different communication cycle. The shuffling algorithm leverages the network parameters (elaborated in section IV.A) which are loaded onto it during the bootstrapping phase. The Shuffling Module works independently of any other module. This means that the Priority List generated by the Shuffling Module need not be synchronized with any other module within the vehicle. The Shuffling Module can be implemented as separate hardware module that can be installed in vehicles, the details of which is outside the scope of this paper.

- SMOD : SMOD is a separate module that performs the arbitration mechanism instead of the CAN. The SMOD is connected to the Shuffling Module, from which it receives the updated priority list during every communication cycle. The priority list received is leveraged to perform the arbitration process of the different ECUs in contention. Furthermore, the SMOD is connected to the CAN, where it relays the dominant bits obtained from the ECUs, which is further broadcast into the entire in-vehicle network. In other words, SMOD acts as an interface between the ECUs and the CAN for sending the bits.

3.2. ASSUMPTIONS

In this paper, we have the following assumptions:

- No insider attack: This means that none of the ECUs within the vehicle is compromised/manipulated.
- Safe Network Shuffling Parameters: The network shuffling parameters used for shuffling the mssage priorities is not disclosed to the attacker through any ECUs within the vehicle or the car manufacturing company.
- Untampered Security Module: The proposed security module introduced/installed within the in-vehicle network cannot be tampered. Furthermore, the shuffling technique is used as a black box within the ECU and the SMOD that is used to generate the new IDs of the ECUs while preserving the relationship between the message priorities originating from the different ECUs.
- No remote connection: Only ECUs and SMOD can connect to the Shuffling Module. Attacker remotely cannot access the Shuffling Module.

- Single Injection Attack: We assume that at any given communication cycle, only one bit can be remotely injected for different arbitration ID bit positions during the arbitration mechanism. In other words, any one message ID can be injected in the CAN for the arbitration mechanism in a given communication cycle.

3.3. THREAT MODEL

In this paper, we assume that the attacker has the following capability:

- Denial of Service: The intention of an attacker is to generate the highest priority message at any given communication cycle to take control of the CAN and deny other legitimate ECUs from sending vital information. The attacker wants to disrupt the normal functioning of the vehicles by forcing the ECU to transit to listening mode by injecting high priority message at every communication cycle.

- Message Snooping: The attacker can snoop into the dominant bits from the CAN at every communication cycle. This is leveraged for the reconnaissance phase, where the attacker gathers the highest arbitration ID at each communication cycle and perform reverse engineering to determine the highest priority arbitration ID at the next communication cycle. The attacker can also perform replay attack by injecting the highest arbitration ID gathered at one communication cycle in a different cycle. The attacker can also snoop into the message payload sent by an ECU. However, such an attack is beyond the scope of this paper.

- Replay Attack: The attacker can replay the information gathered by snooping the CAN (i.e. the winning arbitration ID) from one communication cycle in other subsequent communication cycles. The replay attack in our threat model does not involve replaying the payload captured from an ECU by an attacker.

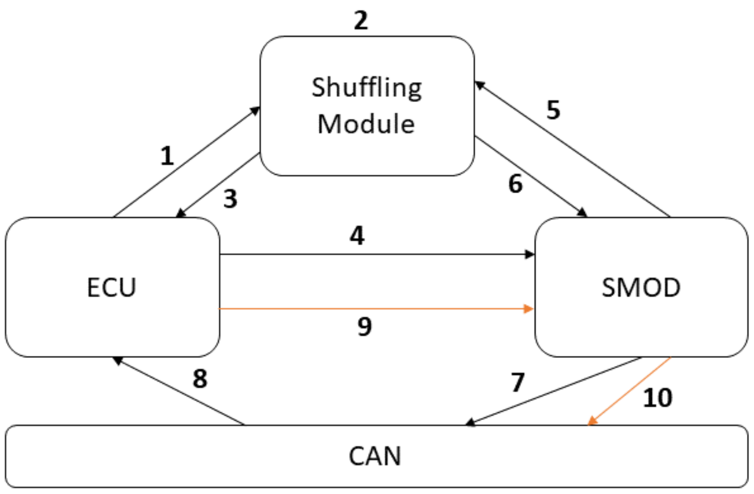


Figure 3. Workflow: **1)** ECU requests for new ID at timestamp, t, **2)** Shuffling Module generates the new IDs of all the ECUs using the shuffling algorithm , **3)** Shuffling Module sends the new ID corresponding to the requested ECU ID, **4)** ECU sends the arbitration ID (new ID obtained in Step 3) to SMOD, **5)** SMOD requests for the current priority List stored in Shuffling Module, **6)** Shuffling Module sends the priority list to SMOD, **7)** SMOD sends the arbitration ID in the CAN (1-bit at a time), **8)** ECU hears the current dominant arbitration bit to decide whether to change in listening mode, **9)** ECU that wins arbitration at timestamp, t sends payload to SMOD, **10)** SMOD forwards payload to CAN that is broadcast within the in-vehicle network

4. PROPOSED MECHANISM

Figure 3 provides an overview of the proposed mechanism where arbitration of the messages originating from the ECUs is delegated to SMOD rather than the CAN. The SMOD is responsible for deciding the ECU that wins the arbitration at a communication cycle, and forwards the payload to the CAN, which is further broadcast within the in-vehicle network. The steps of the proposed mechanism has been detailed below.

4.1. NETWORK PARAMETER GENERATION

During this phase, the Shuffling Module of every vehicle is loaded with the network parameters that is responsible for generating the arbitration ID of the ECU at every communication cycle, T. As network parameters, 2 prime numbers, p and q, are chosen at random along with an evolving factor, α. The purpose of α is to arbitrarily change or evolve the prime numbers that are leveraged for generating the *seed* value, that brings more randomness to the generation of the arbitration ID at a given communication cycle.

Algorithm 11 Arbtration ID shuffling algorithm

```
1: **Input**: $\alpha$, $p_{\text{T-1}}$, $q_{\text{T-1}}$, $\text{ECU}_{\text{T-1}}$ (optional)
2: **Output**: *Priority List* or $\text{ECU}_{\text{T}}$
3: *Priority List*: IDs in decreasing order of priority
4: $p_{\text{T}}, q_{\text{T}}, totalB = \alpha(p_{\text{T-1}}), \alpha(q_{\text{T}}), 11$ or $29$
5: bitS = random number between 1 and $totalB$
6: $\lambda(n) = lcm(p_{\text{T}} - 1, q_{\text{T}} - 1)$
7: $seed = T * \lambda(n)$
8: priority bits = $sample(seed, bitS, totalB)$
9: $\text{ECU}_{\text{count}}$ = total number of unique ECU IDs in vehicle
10: highest priority = Set 1 in priority bits, rest 0
11: *Priority List*.append(highest priority)
12: **for** i in range(1,$\text{ECU}_{\text{count}}$) **do**
13:     *Priority List*.append(highest priority $\oplus$ i)
14: Map $\text{ECU}_{\text{T-1}} \rightarrow \text{ECU}_{\text{T}}$ from *Priority List*
15: **return** *Priority List* or $\text{ECU}_{\text{T}}$
```

During the bootstrapping phase (during the initial setup of Shuffling Module), 2 large distinct prime numbers, $p_{(\text{init})}$ and $q_{(\text{init})}$ along with an evolving factor α is set as communication parameters that is kept private to the Shuffling Module that performs the Shuffling algorithm. At a given communication cycle, the priority list of the message IDs is generated using a seed value, *seed*, generated using equation 2 that leverage the *Carmichael's totient function ($\lambda(n)$)* calculated using equation 1.

$$\lambda(n) = lcm(p_{\text{T}} - 1, q_{\text{T}} - 1) \tag{1}$$

$$seed = T * \lambda(n) \tag{2}$$

where p_{T} and q_{T} is the prime number generated at timestamp, T using α.

Thereafter, the *seed* value is used to determine message ID with the highest priority by setting particular bits (b_0,b_1,...,b_{28}) to high/low, assuming we use 29 bit CAN 2.0B with extended identifiers. The algorithm to generate the priority list is shown in Algorithm 11.

4.2. ECU ID GENERATION AND SENDING ARBITRATION ID

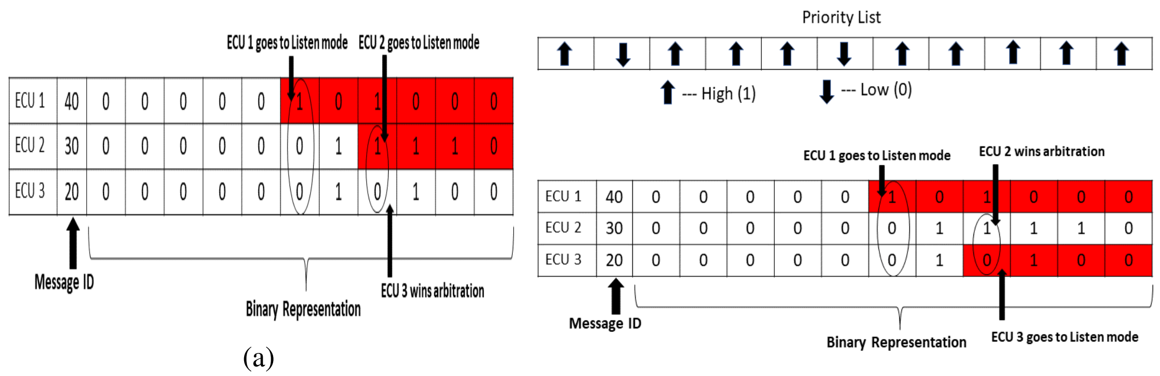
When certain actions are performed on the vehicle, such as when the brake is applied, the associated ECU sends a *ECUIDGeneration* request packet to the Shuffling Module consisting of the following:

$$ECUIDGeneration =< ECU_{\text{T-1}} >$$

where ECU_{T-1} is the current ID of the ECU. On receiving the request, the Shuffling algorithm generates the new *Priority List* and maps the IDs ($\text{ECU}_{\text{T-1}}$) with the IDs of the new *Priority List* (ECU_{T}) as follows: $\text{ECU}_{\text{T-1}}$ in *Priority List*$_{\text{T-1}}$ $\longrightarrow$ ECU_{T} in *Priority List*$_{\text{T}}$

where *Priority List*$_{\text{T-1}}$ and *Priority List*$_{\text{T+1}}$ is the *Priority List* generated at communication cycle $T-1$ (in the previous iteration) and T (at the current iteration) respectively. The Shuffling Algorithm module returns the ECU_{T} to the requesting ECU. Furthermore, the Shufflng module stores the current Priority List generated at communication cycle, T, for sending to the SMOD that indicates the initiation of an arbitration mechanism. This is to be noted that the Priority List is only generated when an ECU sends the *ECUIDGeneration* to the Shuffling Module, which indicates the start of a new communication cycle.

In order to generate the new IDs of the ECUs for the Priority List, the $seed$ is generated using $\lambda(n)$ as shown in equation 1 and 2 (Algorithm 11 lines 4-7). Thereafter it is leveraged to rearrange the priorities of the message IDs at a communication cycle T as shown in Algorithm 11 (lines 9-13). Furthermore, it is important to note that we do not change the priorities of the messages originating from the ECUs. In other words, as commonly agreed by the SAE, if a message originating from ECU_i has higher priority than



(a)

(b)

Figure 4. Arbitration Mechanism: (a) Traditional mechanism and (b) Proposed Mechanism

that originating from ECU_j, it remains so at every communication cycle in our proposed mechanism. However, the arbitration ID and the message priority does not follow a strict relationship (lower message ID means higher priority) unlike the state-of-the-art solutions.

Furthermore, it is also to be noted that the Shuffling Module is the sole module in the proposed mechanism that is leveraged to generate the ECU IDs at every communication cycle. Therefore, α can be designed in any suitable way which is not the main focus of the paper. As there is no synchronization involved between the Shuffling Module and any other module, it does not impact the efficacy of our proposed mechanism.

4.3. ARBITRATION PHASE

We follow the same arbitration process as the state-of-the art standards. The SMOD receives the current *Priority List* from Shuffling Module, when it receives a bit of the arbitration ID from one or more ECUs that are in the arbitration in a communication cycle. If the SMOD receives bits from only one ECUs it directly gets to use the CAN. This is because there is no other ECUs in contention. As a result the bits from the only requested ECU is broadcast into the CAN, which eventually wins the arbitration and sends the payload. However, if the SMOD receives arbitration bits from multiple ECUs, the arbitration process is based on the IDs in the *Priority List*. Therefore, if *n* number of ECUs want to send messages into the CAN, the dominant bit during the arbitration phase is decided as shown in equation 3.

$$dominant = max(ECU_1, ECU_2, ..., ECU_{\mathrm{n}}) \tag{3}$$

where $\mathrm{ECU_i}$ is the bit received from the arbitration ID sent by the $\mathrm{i^{th}}$ ECU in contention and dominant stores the bit value (0/1) that has the highest priority out of bits obtained from the ECUs. The highest priority is decided based on the Priority List obtained from the Shuffling Module

An ECU sends the next arbitration bits only if its previous arbitration bit resembles the value observed in the CAN (i.e dominant value). However, if it does not match, the ECU goes into the listening mode and waits for the next arbitration cycle for sending the message. The difference in arbitration between the proposed mechanism and the traditional (state-of-the-art) mechanism has been highlighted in Figure 4. In Figure 4(b). it is to be noted that the Priority List generated for the communication cycle has high values (1) as dominant bits in positions $0-4, 6-8, 11$ and low values (0) as dominant bits in positions $5, 10$. Therefore, ECU1 goes into listening mode as it is not the dominant bit out of the contending ECUs at the bit position 5. Furthermore, unlike the traditional mechanism (Figure 4(a)), ECU3 transitions into listening mode because it does not have the dominant bit out of the contending ECUs in bit position 4. Thus, ECU2 wins the arbitration even if ECU ID of ECU2 is greater than ECU3. It is also to be noted that ECU1 has a higher ID that ECU2 but loses the arbitration. This shows that we do not have any strict relationship unlike the traditional mechanism, where ECU3 would have won the arbitration (as shown in Figure 4(a)) as it has the lowest arbitration ID.

4.4. MESSAGE SENDING

If the last bit send by an ECU is reflected as a dominant value in the CAN, that ECU wins the arbitration (meaning all other contending ECU has transitioned into listening mode). Under such circumstance, the winning ECU completes the message transfer during the communication cycle by sending the remainder of the message (including the payload) to the SMOD, which is eventually reflected/broadcasted into the CAN. On the other hand, the other contending ECUs that lost the arbitration during the communication cycle , T, waits for receiving the *EOF* bit from the sending ECU. Once received, it reconnects with the Shuffling Module to generate the arbitration ID for the next communication cycle. The Shuffling Module repeats the same process as described previously in section IV(b) and replace the Priority List generated in the previous communication cycle with the current Priority List. Thereafter the same steps are followed for the arbitration mechanism.

5. EVALUATIONS AND EXPERIMENTAL RESULTS

The performance and efficacy of the proposed mechanism has been tested in an Intel core i5-9300H CPU 2.40 GHz workstation, and coded in Python 3.0. We determine the performance of the Shuffling Module with both 11 (CAN 2.0A) and 29 (CAN 2.0B) bit setup while assuming varying number of ECUs in the vehicles, to determine the scalability of the shuffling algorithm.

As seen from Figure 5, it was observed that as the number of message in the contention as well as the priority of those messages decrease, the success rate of the attack increase. For example, if m1 is the only message in contention during a communication cycle and has the least priority (the last element of the Priority List in Algorithm 11) among all the ECUs, the attacker has to simply complement one of the bit out of 11/29 bits. This is because the bits of all other arbitration IDs have a higher priority that m1. Therefore, if the attacker is able to inject one bit complement to m1, the ECU corresponding to m1 will go

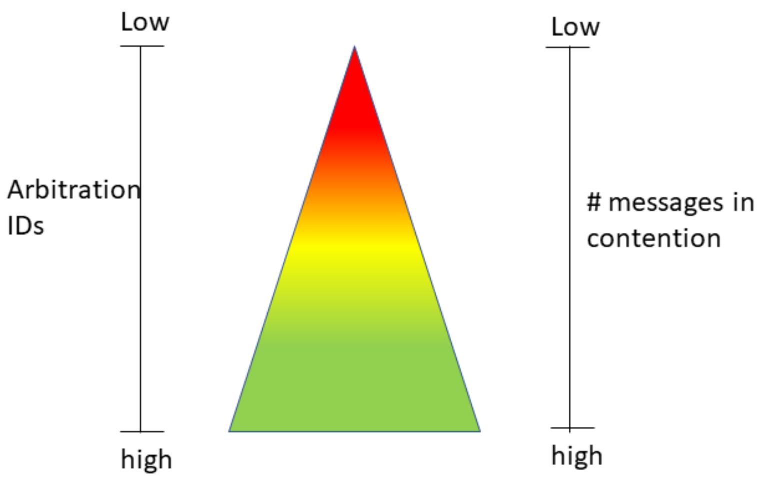


Figure 5. Risk of an attack: **Red**: High risk, **Yellow**: Moderate Risk, **Green**: Low risk

into listening mode. However, if the number of messages in the contention increases, then the other messages have a priority higher than that of m1. In order to successfully attack the system, the attacker has to generate the highest priority among all the messages in the contention, the difficulty of which increases as the number of messages in the contention increases.

We experimented the time taken for Algorithm 11 to generate the Priority list at a give communication cycle, the results of which are shown in Figure 6 . The Shuffling algorithm (Algorithm 11) was tested for both 11 and 29 bit arbitration IDs, and with varying number of ECUs (from 50-200, as the modern day vehicle consists of upto 100 ECUs [23]). It was observed that the time taken for generating the Priority list increased by a small amount even if the number of ECUs in the vehicles increased, suggesting that the proposed mechanism is fairly scalable even if the number of ECUs in the future vehicles increases.

We also determined the frequency of the Shuffling algorithm for the effective working of the proposed mechanism. As shown in Table 1, it can be seen that the success rate of an attack is high when the same Priority List is used at different communication cycle. Under such circumstance, the attacker has to gather the highest arbitration ID at a given

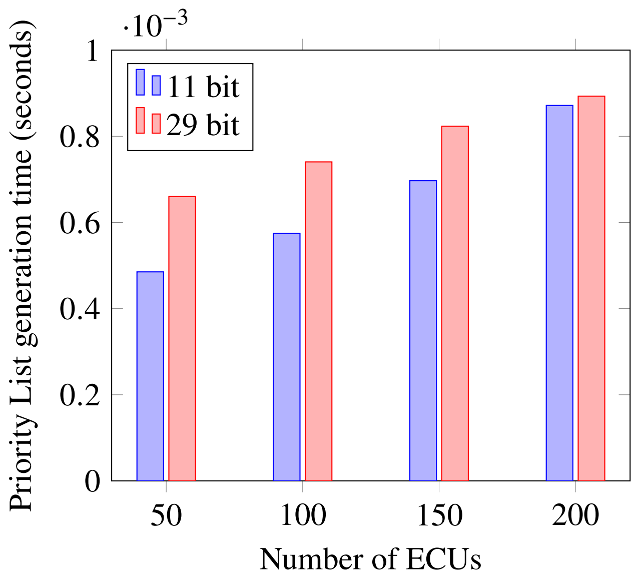


Figure 6. Average time (out of 20 for each configuration) taken for the Priority list generation using Algorithm 11

communication channel and replay it in the next communication. However, it is to be noted that at a given communication channel, the attacker is able to gather the highest priority among the arbitration IDs in contention, which may or may not be the highest priority ID among all the ECUs. Nevertheless, the attacker can continue gathering information for a larger time, and eventually get the highest priority ID as the same Priority List is generated at every communication cycle. It was also observed that the success rate of an attack increases if the Priority List is generated at regular intervals (and not every communication cycle). To illustrate such scenario, let us assume that a Priority List is generated at time interval, T1, and is changed every δt seconds. Under such circumstance, the attacker can snoop into the arbitration IDs every δt seconds to reverse engineer the priority of the IDs. Thus the attack success can be High if the attacker finds the highest arbitration ID at a time $<< T1 + \delta$t seconds. However, as stated earlier, since the attacker can only know the highest arbitration ID among the IDs in contention in a given communication cycle, the attack success is Moderate if shuffled at regular intervals. Nevertheless, if the Priority List is generated at every communication cycle (i.e., when any ECU want to sent data into the CAN), it adds randomness into the time of generation of Priority List, which decreases the attack success within the system.

Table 1. Attack success under different Shuffling strategy

Shuffling Strategy	**Attack Success**
Using Same Priority List	High
Generating priority List at regular time intervals	Moderate - High
Generating Priority List every communication cycle	Low

We also analyzed the performance against DoS during the arbitration phase with some of the existing researches as shown in Table 2. A major drawback of the proposed approach in [24] and [25] is that it assumes that the attacker either is unaware of the sequence of the message within the CAN (as in [24]), or the attacker infinitely injects the lowest priority within the CAN (as in [25]). However, if the attacker gathers information of the message/arbitration IDs active in the CAN by snooping into the dominant bits in CAN, it can reverse engineer to gather the sequence of the message IDs, and inject it within the CAN to successfully execute a DoS. Also, instead of continuously injecting the high priority message in the CAN, if the attacker knows the high priority message and injects it at an acceptable frequency delay, it can still win the arbitration, as the messages originating from the ECUs may not follow a regular frequency. Under such a scenario, the attacker can perform DoS in a given communication cycle if it has the highest ID among the contending ECUs during the communication cycle. Furthermore, an important assumption of the existing models assumes that the entire arbitration ID is available during the detection phase, whereas in reality only a single bit at a time is available in the CAN. Therefore, the model has to wait for the entire arbitration bits to be sent to the CAN, which increases the active time of the ECUs (as the ECUs can transit to listening mode after seeing the dominant bit).

Table 2. Performance Comparison against DoS

Katragadda et al. [24]	✗
Lee et al. [25]	✗
Humayed et al. [22]	✓
CanSafe	✓

On the other hand, the authors in [22] proposed a deterministic finite automation (DFA) based mechanism that is resilient to DoS attack during the arbitration phase. However, the proposed mechanism assumes that the attacker does not know the list of legitimate ECU IDs in the vehicle. Considering such an assumption, the efficacy of the mechanism can still degrade if the attacker sniffs into the CAN data for a long time (as reconnaissance) to reverse engineer one or more legitimate ECU IDs. As a result, such information/ID gathered can be replayed at a different communication cycle to disrupt the arbitration mechanism. This is because the list containing the IDs of the ECUs remains the same, and the attacker is able to inject a valid ECU ID that belongs to the whitelist. We compared the probability of an attacker successfully winning the arbitration under different scenarios between *CanSafe* and [22]. In Table 3, Scenario 1 is the condition where the attacker has the ECU IDs in the white list that is maintained by the authors in [22]. Under such circumstances, the attacker can always win the arbitration by injecting the lowest ID from the white list at every communication cycle. However, the probability decreases to a maximum 0.5 if the attacker does not have the white list (i.e. the attacker injects the last arbitration bit which has a 50% chance of winning). However, it is to be noted that under such circumstances, the attacker can easily perform reconnaissance, and reverse engineer the entire white list by collecting

Table 3. Arbitration Attack Success

Humayed et al. [22] – Scenario-1	1
Humayed et al. [22] – Scenario-2	≤ 0.5
CanSafe	≤ 0.5

the winning arbitration IDs during a communication cycle. From the ECU IDs gathered from multiple communication cycles, the entire white list(which remains the same at all communication cycles) can be generated. In *CanSafe* the probability is also a maximum of 0.5, when the attacker has a 50% chance of injecting the last arbitration bit correctly. However, since in our mechanism, the ECU IDs and the priority list is changed randomly at each communication cycle, the attacker cannot attack the CAN arbitration mechanism with a probability of 1.

6. CANSAFE ANALYSIS

In this subsection, we analyze the performance of the proposed mechanism against the different threats posed by an attacker.

6.1. BRUTE-FORCE ATTACK

In this attack, the attacker constantly injects the lowest message ID (highest priority as per the state-of-the-art specification) #0000 to perform DoS within the in-vehicle network. As the proposed mecanism reshuffles the priority of the messages at every iteration, the probability of the brute-force attack to be successful is very low (represented below as P(success) in equation 4).

$$P(success) <= \frac{1}{2^{n}} \tag{4}$$

where n is the number of bits used to represent the arbitration ID (11 bits for CAN 2.0A and 29 bits for CAN 2.0B).

However, it is to be noted that even if #0000 is the highest priority message at a time, it may happen that the ECU with the actual #0000 sends a message. In the event of such a scenario, the arbitration mechanism fails due to collision. In the next arbitration cycle, #0000 is mapped to some other ID which is known by the authentic ECU in the contention but not by the attacker. Hence, the attack fails under such circumstance. Thus, it can be concluded that the intention of the attacker to inject the lowest priority message within the network is successful only if random shuffling yields #0000 as the highest priority message and the authentic ECU associated with the ID does not send any message (i.e. no message collision).

6.2. ATTACKER RECONNAISSANCE AND REPLAY ATTACK

In this attack, the attacker gathers the information at one cycle and leverage it for injecting message into the next cycle. The information that can be gathered by the attacker includes the arbitration ID that won the contention in the previous cycle, and the message sent by the ECU. At this stage the attacker may perform 2 kinds of attack : 1) Naive Attack 2) Reverse Engineering.

In the Naive attack, the attacker performs replay attack by injecting the highest arbitration ID gathered in the previous cycle. Under such circumstance the attack can be successful if one of the condition holds:

- Case 1: The arbitration ID gathered (which may not have been the highest priority ID at the previous cycle but was the highest among the IDs in contention) is the highest priority ID among all the ECU IDs in the vehicle or among the IDs in contention (which means that the random shuffling generates the same Priority List, which is highly unlikely) at the next cycle.

- Case 2: The attacker's injected message is the only message in contention.

For Case 1, the chances that the arbitration ID that won in the previous cycle becoming the highest priority among all the ECUs in the next cycle can be represented by equation 4, i.e., all the bits has to be exactly the same. Nevertheless, it is to be noted that such the attacks in Case 1 can be handled by simple check conditions that does not allow a ID that won the arbitration in a cycle to become the highest priority ID in the next cycle or no same priority list at consecutive communication cycles. As the priority list is generated only by the Shuffling Module, such a check condition does not affect the overall communication flow of the mechanism.

For Case 2, this is not a DoS attack but is more of a message injection attack, which is outside the scope of this paper.

In case of a Reverse Engineering attack, the attacker gathers the information of different cycle and tries to find out the random bits (Algorithm 11, line 8) that determines the priority list. However, it is to be noted that at any given cycle, the attacker has to try out 2048 (for 11 bits) and $\approx$ 500 million (for 29 bits) to figure out the random factor. As Shuffling Module is the only module that is generating the priority list (with no synchronization with other modules), this proposed mechanism can exploit any relationship between the random factor used in different communication cycles that need not be a linear relationship. Such a independence of the Shuffling Module obfuscates the random factor from an attacker at different communication cycle.

6.3. BIT INFERRING BY AN ATTACKER

The connection between the ECU and the Shuffling Module is secure, meaning it cannot be intercepted by an attacker. Further, the attacker cannot connect to the Shuffling Module. As a result of these constraints, the attacker cannot determine the ECU ID of an ECU at different communication cycle. As a single bit is sent by every ECU to the SMOD, that is reflected at the CAN, the attacker has no way to determine the bits sent by the ECUs in contention, in advance. As a result, the attacker cannot infer the dominant bit at a given time, without listening to it from the CAN during a communication cycle.

However, as it is mentioned in section 5(Figure 5), the attack success of an attacker increases if the number of messages in the contention is less and the priority of the message is also less. It may happen that the attacker does not inject any bit until the last bit of the arbitration ID. Under such circumstance the probability of an attack is maximum (=0.5). This is because the last bit of the arbitration ID can be 0 or 1. This means that the dominant bit can be either 0 or 1. So the attacker has to sent either 0 or 1 into the CAN. If the attacker sends 0, a contending ECU sends 1 and the dominant bit is 1, then the attacker loses the arbitration. However, if the attacker sends 1 and the contending ECU sends 0, then the attacker wins the arbitration and the the ECUs go into sleep mode. Thus, the attack success of an attacker is the maximum when only the last bit of the arbitration ID is injected into the CAN.

6.4. DYNAMIC ECU ID PREVENTS TRACKING

As the ECUs use different dynamic arbitration ID which is allocated to it by the Shuffling Module, any attacker that wants to compromise an ECU, or masquerade to be some other legitimate ECU should be able to generate the arbitration ID of the ECU in a given communication cycle. As the number of ECUs, is around 100 in a modern day vehicle, the unique arbitration IDs in a given communication cycle is a subset of the number of possible arbitration IDs at every communication cycle (2^{11} for CAN 2.0A and 2^{29} in our proposed mechanism). Thus, it is difficult for an attacker to target a specific ECU as it has to correctly determine the ECU ID in the given communication cycle. Furthermore, as the proposed mechanism is also resilient to replay attack, it shows that the attacker cannot replay a captured ECU ID from an ECU at one communication cycle and use it to inject message in a different communication cycle. Thus, the proposed mechanism prevents masquerading attack against any ECU. Furthermore, as the ECU IDs are dynamic, it is also difficult for an attacker to associate a payload with any ECU, as it may happen that an arbitration/ECU ID belonging to one ECU in a communication cycle may be allocated to some other ECU in a different communication cycle. A naive solution to detect any aberration in a data is to design a simple anomaly detection system which will determine if any ECU gives anomalous data in different communication cycle (differing by a certain threshold), which is outside the scope of this paper.

6.5. MULTIPLE INJECTION BY ATTACKER/S

Our proposed mechanism is able to withstand DoS attack as long as there is a single injection inside the CAN bus, i.e. single message ID is injected at every communication cycle. However, it is to be noted, that the attacker can perform a more sophisticated attack by injecting 2 bits in a communication cycle. This may happen if a single attacker is able to perform 2 injection attacks within the CAN bus, or more than one attacker colludes to inject 0 and 1 simultaneously for each bit position. On the advent of such a scenario, the attacker can effectively effectively deny all legitimate ECUs by winning the arbitration if the 2 injected bits are 0 and 1 in the last bit position during the arbitration of a communication cycle. Under such circumstance, the attacker will either win the arbitration or result in a collision. Such an attack is not addressed in our proposed mechanism.

7. CONCLUSION

In this paper, we have proposed a new shuffling algorithm for dynamically generating the priority IDs of the ECUs within in-vehicle networks. The proposed mechanism outperforms the traditional mechanism (lowest arbitration ID- highest priority in CAN) used for the arbitration process. We leverage the MTD based shuffling technique that nullifies the reconnaissance phase of an attacker and provides resiliency against DoS, as the attacker cannot leverage the information gathered in one communication cycle and generate the highest priority message in the next communication cycle. We also showed that the proposed shuffling algorithm has a fairly constant time requirement for both 11 and 29 bits CAN architecture. We also performed a detailed analysis of the proposed mechanism and showed the possibilities of various conditions under which an attacker can to bypass/break the shuffling mechanism. Thus, the proposed mechanism enhances the security of the existing CAN specification and preserve the priority of the messages originating from the ECUs even if their arbitration IDs do not follow a strict relationship. For the future, we would like to analyze and enhance our proposed mechanism to handle multiple injection attacks by remote attackers.

REFERENCES

[1] H. M. Song and H. K. Kim, "Self-supervised anomaly detection for in-vehicle network using noised pseudo normal data," *IEEE Transactions on Vehicular Technology*, vol. 70, no. 2, pp. 1098–1108, 2021.

[2] S. Nakamura, K. Takeuchi, H. Kashima, T. Kishikawa, T. Ushio, T. Haga, and T. Sasaki, "In-vehicle network attack detection across vehicle models: A supervised-unsupervised hybrid approach," in *2021 IEEE International Intelligent Transportation Systems Conference (ITSC)*, pp. 1286–1291, IEEE, 2021.

[3] T. Huybrechts, Y. Vanommeslaeghe, D. Blontrock, G. Van Barel, and P. Hellinckx, "Automatic reverse engineering of can bus data using machine learning techniques," in *International Conference on P2P, Parallel, Grid, Cloud and Internet Computing*, pp. 751–761, Springer, 2017.

[4] C. Miller and C. Valasek, "Adventures in automotive networks and control units," *Def Con*, vol. 21, no. 260-264, pp. 15–31, 2013.

[5] R. Zhuang, S. A. DeLoach, and X. Ou, "Towards a theory of moving target defense," in *Proceedings of the First ACM Workshop on Moving Target Defense*, pp. 31–40, 2014.

[6] J.-H. Cho, D. P. Sharma, H. Alavizadeh, S. Yoon, N. Ben-Asher, T. J. Moore, D. S. Kim, H. Lim, and F. F. Nelson, "Toward proactive, adaptive defense: A survey on moving target defense," *IEEE Communications Surveys & Tutorials*, vol. 22, no. 1, pp. 709–745, 2020.

[7] L. B. Othmane, L. Dhulipala, M. Abdelkhalek, N. Multari, and M. Govindarasu, "On the performance of detecting injection of fabricated messages into the can bus," *IEEE Transactions on Dependable and Secure Computing*, 2020.

[8] H. Narasimhan, R. Vinayakumar, and N. Mohammad, "Unsupervised deep learning approach for in-vehicle intrusion detection system," *IEEE Consumer Electronics Magazine*, 2021.

[9] H. M. Song, J. Woo, and H. K. Kim, "In-vehicle network intrusion detection using deep convolutional neural network," *Vehicular Communications*, vol. 21, p. 100198, 2020.

[10] M. Hanselmann, T. Strauss, K. Dormann, and H. Ulmer, "Canet: An unsupervised intrusion detection system for high dimensional can bus data," *IEEE Access*, vol. 8, pp. 58194–58205, 2020.

[11] V. Tanksale, "Anomaly detection for controller area networks using long short-term memory," *IEEE Open Journal of Intelligent Transportation Systems*, vol. 1, pp. 253–265, 2020.

[12] D. Shi, M. Xu, T. Wu, and L. Kou, "Intrusion detecting system based on temporal convolutional network for in-vehicle can networks," *Mobile Information Systems*, vol. 2021, 2021.

[13] M. Jedh, L. b. Othmane, N. Ahmed, and B. Bhargava, "Detection of message injection attacks onto the can bus using similarity of successive messages-sequence graphs," *arXiv preprint arXiv:2104.03763*, 2021.

[14] R. Islam, R. U. D. Refat, S. M. Yerram, and H. Malik, "Graph-based intrusion detection system for controller area networks," *IEEE Transactions on Intelligent Transportation Systems*, 2020.

[15] K.-T. Cho and K. G. Shin, "Fingerprinting electronic control units for vehicle intrusion detection," in *25th {USENIX} Security Symposium ({USENIX} Security 16)*, pp. 911–927, 2016.

[16] A. Taylor, N. Japkowicz, and S. Leblanc, "Frequency-based anomaly detection for the automotive can bus," in *2015 World Congress on Industrial Control Systems Security (WCICSS)*, pp. 45–49, IEEE, 2015.

[17] H. M. Song, H. R. Kim, and H. K. Kim, "Intrusion detection system based on the analysis of time intervals of can messages for in-vehicle network," in *2016 international conference on information networking (ICOIN)*, pp. 63–68, IEEE, 2016.

[18] M. Marchetti, D. Stabili, A. Guido, and M. Colajanni, "Evaluation of anomaly detection for in-vehicle networks through information-theoretic algorithms," in *2016 IEEE 2nd International Forum on Research and Technologies for Society and Industry Leveraging a better tomorrow (RTSI)*, pp. 1–6, IEEE, 2016.

[19] M. Müter and N. Asaj, "Entropy-based anomaly detection for in-vehicle networks," in *2011 IEEE Intelligent Vehicles Symposium (IV)*, pp. 1110–1115, IEEE, 2011.

[20] S. Yoon, J.-H. Cho, D. S. Kim, T. J. Moore, F. F. Nelson, H. Lim, N. Leslie, and C. Kamhoua, "Moving target defense for in-vehicle software-defined networking: Ip shuffling in network slicing with multiagent deep reinforcement learning," in *Artificial Intelligence and Machine Learning for Multi-Domain Operations Applications II*, vol. 11413, p. 114131U, International Society for Optics and Photonics, 2020.

[21] S. Woo, D. Moon, T.-Y. Youn, Y. Lee, and Y. Kim, "Can id shuffling technique (cist): Moving target defense strategy for protecting in-vehicle can," *IEEE Access*, vol. 7, pp. 15521–15536, 2019.

[22] A. Humayed, F. Li, J. Lin, and B. Luo, "Cansentry: Securing can-based cyber-physical systems against denial and spoofing attacks," in *European Symposium on Research in Computer Security*, pp. 153–173, Springer, 2020.

[23] S. Hamsini and M. Kathiresh, "Automotive safety systems," in *Automotive Embedded Systems*, pp. 1–18, Springer, 2021.

[24] S. Katragadda, P. J. Darby, A. Roche, and R. Gottumukkala, "Detecting low-rate replay-based injection attacks on in-vehicle networks," *IEEE Access*, vol. 8, pp. 54979–54993, 2020.

[25] H. Lee, S. H. Jeong, and H. K. Kim, "Otids: A novel intrusion detection system for in-vehicle network by using remote frame," in *2017 15th Annual Conference on Privacy, Security and Trust (PST)*, pp. 57–5709, IEEE, 2017.

SECTION

3. CONCLUSION AND FUTURE WORK

In our research, we have been able to develop algorithms for detecting the malicious vehicles, and accurately determining the traffic scenario from the ROI even under the influence of the majority of malicious vehicles within the network. We worked on traffic event detection and validation as well as malicious vehicle identification under secured message dissemination in a V2X communication in VANET. The model proposed in Paper I leverage the centralized architecture integrated with PKI schemes for validating the traffic scenario of a region, as well as detecting the malicious vehicles. An improved version of Section 3 has been proposed in Paper II, that leverages a distributed concept for detecting the malicious vehicles and the traffic scenario of a region without any need for a centralized entity. In Paper III, a blockchain-assisted distributed solution is proposed that improves the solution proposed in Paper II with a better degree of transparency at the vehicular plane. Furthermore, a resilient algorithm for securing the arbitration mechanism against DoS attacks within the in-vehicle network is proposed in Paper IV that significantly improves the security of the state-of-the-art arbitration mechanism as defined in the CAN specification. However, there exist challenges with the proposed solutions in accommodating real-time decision-making scenarios for the vehicles such as changing lanes or steering the wheel. Hereafter, we plan to focus on designing an intelligent IDS that would be deployed within the CAN bus of the vehicle and would be effective in identifying any outlier or fabricated information that is received through a V2X communication under the influence of minimum-to-no infrastructure.

For the intelligent IDS within the CAN bus, our goal is to analyze the BSM information that is received from the nearby vehicles nearby together with the current state of the sensors and the actuators to determine the feasibility of the information that is present in the CAN of the vehicle. To determine any intrusion, we plan to analyze the information obtained from the ECUs of the vehicle. For any information sent from one ECU to the other ECU with the in-vehicle network, we will perform a feasibility study of the sender ECU information based on the timestamp of the information when the message was generated, the timestamp of the last message sent from the sender ECU to the target ECU, the content of the message, the corresponding impact of the target ECU to the last sent message from the sender ECU, and many other factors. However, due to the limitations in the computational capability of the CAN bus, and the time constraint for the decision making, a traditional authentication algorithm cannot be incorporated within the CAN bus to authenticate one ECU to the other ECU as that would add a significant amount of delay to the solution, and the CAN may not have the computational capability to perform heavyweight computations. Thus, our goal is to design lightweight authentication algorithms that adhere to the above challenges.

Also, for the in-vehicle networks arbitration mechanism, our proposed solution can handle DoS attacks under the influence of a single injection. However, the efficacy of the proposed mechanism reduces under the influence of multiple injection attacks. Thus, in the future, our goal is to design a strategy that can withstand DoS attacks even under the influence of multiple injection attacks.

REFERENCES

[1] Hamssa Hasrouny, Abed Ellatif Samhat, Carole Bassil, and Anis Laouiti. Vanet security challenges and solutions: A survey. *Vehicular Communications*, 7:7–20, 2017.

[2] Hyun Min Song and Huy Kang Kim. Self-supervised anomaly detection for in-vehicle network using noised pseudo normal data. *IEEE Transactions on Vehicular Technology*, 70(2):1098–1108, 2021.

[3] Shu Nakamura, Koh Takeuchi, Hisashi Kashima, Takeshi Kishikawa, Takashi Ushio, Tomoyuki Haga, and Takamitsu Sasaki. In-vehicle network attack detection across vehicle models: A supervised-unsupervised hybrid approach. In *2021 IEEE International Intelligent Transportation Systems Conference (ITSC)*, pages 1286–1291. IEEE, 2021.

[4] Mohammadreza Bachari Lafte, Omid Jafarzad, and Naimeh Mousavi Ghahfarokhi. International navigation rules governing the unmanned vessels. *Research in Marine Sciences*, 3(2):329–341, 2018.

[5] Kamini Kamini and Rakesh Kumar. Vanet parameters and applications: A review. *Global Journal of Computer Science and Technology*, 2010.

[6] Sherali Zeadally, Ray Hunt, Yuh-Shyan Chen, Angela Irwin, and Aamir Hassan. Vehicular ad hoc networks (vanets): status, results, and challenges. *Telecommunication Systems*, 50(4):217–241, 2012.

[7] Thomas Huybrechts, Yon Vanommeslaeghe, Dries Blontrock, Gregory Van Barel, and Peter Hellinckx. Automatic reverse engineering of can bus data using machine learning techniques. In *International Conference on P2P, Parallel, Grid, Cloud and Internet Computing*, pages 751–761. Springer, 2017.

[8] Charlie Miller and Chris Valasek. Adventures in automotive networks and control units. *Def Con*, 21(260-264):15–31, 2013.

[9] Qin Li, Amizah Malip, Keith M Martin, Siaw-Lynn Ng, and Jie Zhang. A reputation-based announcement scheme for vanets. *IEEE Transactions on Vehicular Technology*, 61(9):4095–4108, 2012.

[10] Shereen A Taie and Sanaa Taha. A novel secured traffic monitoring system for vanet. In *2017 IEEE International Conference on Pervasive Computing and Communications Workshops (PerCom Workshops)*, pages 176–182. IEEE, 2017.

[11] Yujue Wang, Yong Ding, Qianhong Wu, Yongzhuang Wei, Bo Qin, and Huiyong Wang. Privacy-preserving cloud-based road condition monitoring with source authentication in vanets. *IEEE Transactions on Information Forensics and Security*, 14 (7):1779–1790, 2018.

[12] Pranav Kumar Singh, Roshan Singh, Sunit Kumar Nandi, Kayhan Zrar Ghafoor, Danda B Rawat, and Sukumar Nandi. Blockchain-based adaptive trust management in internet of vehicles using smart contract. *IEEE Transactions on Intelligent Transportation Systems*, 2020.

[13] Zhe Yang, Kan Yang, Lei Lei, Kan Zheng, and Victor CM Leung. Blockchain-based decentralized trust management in vehicular networks. *IEEE Internet of Things Journal*, 2018.

[14] Zhaojun Lu, Wenchao Liu, Qian Wang, Gang Qu, and Zhenglin Liu. A privacy-preserving trust model based on blockchain for vanets. *IEEE Access*, 6:45655–45664, 2018.

[15] Xumin Huang, Rong Yu, Jiawen Kang, and Yan Zhang. Distributed reputation management for secure and efficient vehicular edge computing and networks. *IEEE Access*, 5:25408–25420, 2017.

[16] Hakima Khelifi, Senlin Luo, Boubakr Nour, Hassine Moungla, Syed Hassan Ahmed, and Mohsen Guizani. A blockchain-based architecture for secure vehicular named data networks. *Computers & Electrical Engineering*, 86:106715, 2020.

[17] Uzair Javaid, Muhammad Naveed Aman, and Biplab Sikdar. Drivman: Driving trust management and data sharing in vanets with blockchain and smart contracts. In *2019 IEEE 89th Vehicular Technology Conference (VTC2019-Spring)*, pages 1–5. IEEE, 2019.

[18] Yao-Tsung Yang, Li-Der Chou, Chia-Wei Tseng, Fan-Hsun Tseng, and Chien-Chang Liu. Blockchain-based traffic event validation and trust verification for vanets. *IEEE Access*, 7:30868–30877, 2019.

[19] Siri Guleng, Celimuge Wu, Xianfu Chen, Xiaoyan Wang, Tsutomu Yoshinaga, and Yusheng Ji. Decentralized trust evaluation in vehicular internet of things. *IEEE Access*, 7:15980–15988, 2019.

[20] Lotfi Ben Othmane, Lalitha Dhulipala, Moataz Abdelkhalek, Nicholas Multari, and Manimaran Govindarasu. On the performance of detecting injection of fabricated messages into the can bus. *IEEE Transactions on Dependable and Secure Computing*, 2020.

[21] Harini Narasimhan, R Vinayakumar, and Nazeeruddin Mohammad. Unsupervised deep learning approach for in-vehicle intrusion detection system. *IEEE Consumer Electronics Magazine*, 2021.

[22] Hyun Min Song, Jiyoung Woo, and Huy Kang Kim. In-vehicle network intrusion detection using deep convolutional neural network. *Vehicular Communications*, 21: 100198, 2020.

[23] Markus Hanselmann, Thilo Strauss, Katharina Dormann, and Holger Ulmer. Canet: An unsupervised intrusion detection system for high dimensional can bus data. *IEEE Access*, 8:58194–58205, 2020.

[24] Vinayak Tanksale. Anomaly detection for controller area networks using long short-term memory. *IEEE Open Journal of Intelligent Transportation Systems*, 1:253–265, 2020.

[25] Dongxian Shi, Ming Xu, Ting Wu, and Liang Kou. Intrusion detecting system based on temporal convolutional network for in-vehicle can networks. *Mobile Information Systems*, 2021, 2021.

[26] Mubark Jedh, Lotfi ben Othmane, Noor Ahmed, and Bharat Bhargava. Detection of message injection attacks onto the can bus using similarity of successive messages-sequence graphs. *arXiv preprint arXiv:2104.03763*, 2021.

[27] Riadul Islam, Rafi Ud Daula Refat, Sai Manikanta Yerram, and Hafiz Malik. Graph-based intrusion detection system for controller area networks. *IEEE Transactions on Intelligent Transportation Systems*, 2020.

[28] Kyong-Tak Cho and Kang G Shin. Fingerprinting electronic control units for vehicle intrusion detection. In *25th {USENIX} Security Symposium ({USENIX} Security 16)*, pages 911–927, 2016.

[29] Adrian Taylor, Nathalie Japkowicz, and Sylvain Leblanc. Frequency-based anomaly detection for the automotive can bus. In *2015 World Congress on Industrial Control Systems Security (WCICSS)*, pages 45–49. IEEE, 2015.

[30] Hyun Min Song, Ha Rang Kim, and Huy Kang Kim. Intrusion detection system based on the analysis of time intervals of can messages for in-vehicle network. In *2016 international conference on information networking (ICOIN)*, pages 63–68. IEEE, 2016.

[31] Mirco Marchetti, Dario Stabili, Alessandro Guido, and Michele Colajanni. Evaluation of anomaly detection for in-vehicle networks through information-theoretic algorithms. In *2016 IEEE 2nd International Forum on Research and Technologies for Society and Industry Leveraging a better tomorrow (RTSI)*, pages 1–6. IEEE, 2016.

[32] Michael Müter and Naim Asaj. Entropy-based anomaly detection for in-vehicle networks. In *2011 IEEE Intelligent Vehicles Symposium (IV)*, pages 1110–1115. IEEE, 2011.

[33] Seunghyun Yoon, Jin-Hee Cho, Dong Seong Kim, Terrence J Moore, Frederica F Nelson, Hyuk Lim, Nandi Leslie, and Charles Kamhoua. Moving target defense for in-vehicle software-defined networking: Ip shuffling in network slicing with multi-agent deep reinforcement learning. In *Artificial Intelligence and Machine Learning for Multi-Domain Operations Applications II*, volume 11413, page 114131U. International Society for Optics and Photonics, 2020.

[34] Samuel Woo, Daesung Moon, Taek-Young Youn, Yousik Lee, and Yongeun Kim. Can id shuffling technique (cist): Moving target defense strategy for protecting in-vehicle can. *IEEE Access*, 7:15521–15536, 2019.

[35] Abdulmalik Humayed, Fengjun Li, Jingqiang Lin, and Bo Luo. Cansentry: Securing can-based cyber-physical systems against denial and spoofing attacks. In *European Symposium on Research in Computer Security*, pages 153–173. Springer, 2020.

[36] D Manivannan, Shafika Showkat Moni, and Sherali Zeadally. Secure authentication and privacy-preserving techniques in vehicular ad-hoc networks (vanets). *Vehicular Communications*, page 100247, 2020.

[37] Jun Shao, Xiaodong Lin, Rongxing Lu, and Cong Zuo. A threshold anonymous authentication protocol for vanets. *IEEE Transactions on vehicular technology*, 65(3): 1711–1720, 2015.

[38] Philippe Golle, Dan Greene, and Jessica Staddon. Detecting and correcting malicious data in vanets. In *Proceedings of the 1st ACM international workshop on Vehicular ad hoc networks*, pages 29–37. ACM, 2004.

[39] Andy Greenberg. Hackers remotely kill a jeep on the highway—with me in it. *Wired*, 7:21, 2015.

[40] William Whyte, André Weimerskirch, Virendra Kumar, and Thorsten Hehn. A security credential management system for v2v communications. In *2013 IEEE Vehicular Networking Conference*, pages 1–8. IEEE, 2013.

[41] Zhen Huang, Sushmita Ruj, Marcos A Cavenaghi, Milos Stojmenovic, and Amiya Nayak. A social network approach to trust management in vanets. *Peer-to-Peer Networking and Applications*, 7(3):229–242, 2014.

[42] Liehuang Zhu, Chuan Zhang, Chang Xu, Xiaojiang Du, Nadra Guizani, and Kashif Sharif. Traffic monitoring in self-organizing vanets: A privacy-preserving mechanism for speed collection and analysis. *IEEE Wireless Communications*, 26(6):18–23, 2019.

[43] Hamssa Hasrouny, Abed Ellatif Samhat, Carole Bassil, and Anis Laouiti. Trust model for secure group leader-based communications in vanet. *Wireless Networks*, pages 1–23, 2018.

[44] Joan Daemen and Vincent Rijmen. The block cipher rijndael. In *International Conference on Smart Card Research and Advanced Applications*, pages 277–284. Springer, 1998.

[45] Diaa Salama Abd Elminaam, Hatem Mohamed Abdual-Kader, and Mohiy Mohamed Hadhoud. Evaluating the performance of symmetric encryption algorithms. *IJ Network Security*, 10(3):216–222, 2010.

[46] Katja Schmidt-Samoa. A new rabin-type trapdoor permutation equivalent to factoring. *Electronic Notes in Theoretical Computer Science*, 157(3):79–94, 2006.

[47] Ronald L Rivest, Adi Shamir, and Leonard Adleman. A method for obtaining digital signatures and public-key cryptosystems. *Communications of the ACM*, 21(2):120–126, 1978.

[48] Michael O Rabin. Digitalized signatures and public-key functions as intractable as factorization. Technical report, Massachusetts Inst of Tech Cambridge Lab for Computer Science, 1979.

[49] Neal Koblitz. Elliptic curve cryptosystems. *Mathematics of computation*, 48(177): 203–209, 1987.

[50] Krists Magons. Applications and benefits of elliptic curve cryptography. In *SOFSEM (Student Research Forum Papers/Posters)*, pages 32–42, 2016.

[51] Taher ElGamal. A public key cryptosystem and a signature scheme based on discrete logarithms. *IEEE transactions on information theory*, 31(4):469–472, 1985.

[52] David W Kravitz. Digital signature algorithm, July 27 1993. US Patent 5,231,668.

[53] Michael Behrisch, Laura Bieker, Jakob Erdmann, and Daniel Krajzewicz. Sumo–simulation of urban mobility: an overview. In *Proceedings of SIMUL 2011, The Third International Conference on Advances in System Simulation*. ThinkMind, 2011.

[54] Morris J Dworkin. Sha-3 standard: Permutation-based hash and extendable-output functions. Technical report, 2015.

[55] Kazumaro Aoki and Yu Sasaki. Preimage attacks on one-block md4, 63-step md5 and more. In *International workshop on selected areas in cryptography*, pages 103–119. Springer, 2008.

[56] Hongbo Yu, Gaoli Wang, Guoyan Zhang, and Xiaoyun Wang. The second-preimage attack on md4. In *International Conference on Cryptology and Network Security*, pages 1–12. Springer, 2005.

[57] Kevin S McCurley. The discrete logarithm problem. In *AMS Proc. Symp. Appl. Math*, volume 42, pages 49–74, 1990.

[58] Mohit Arora. How secure is aes against brute force attacks. *EE Times*, 5(7), 2012.

[59] Andrey Bogdanov, Dmitry Khovratovich, and Christian Rechberger. Biclique cryptanalysis of the full aes. In *International Conference on the Theory and Application of Cryptology and Information Security*, pages 344–371. Springer, 2011.

[60] Mani Amoozadeh, Hui Deng, Chen-Nee Chuah, H Michael Zhang, and Dipak Ghosal. Platoon management with cooperative adaptive cruise control enabled by vanet. *Vehicular communications*, 2(2):110–123, 2015.

[61] Rui Zhuang, Scott A DeLoach, and Xinming Ou. Towards a theory of moving target defense. In *Proceedings of the First ACM Workshop on Moving Target Defense*, pages 31–40, 2014.

[62] Jin-Hee Cho, Dilli P Sharma, Hooman Alavizadeh, Seunghyun Yoon, Noam Ben-Asher, Terrence J Moore, Dong Seong Kim, Hyuk Lim, and Frederica F Nelson. Toward proactive, adaptive defense: A survey on moving target defense. *IEEE Communications Surveys & Tutorials*, 22(1):709–745, 2020.

[63] S Hamsini and M Kathiresh. Automotive safety systems. In *Automotive Embedded Systems*, pages 1–18. Springer, 2021.

[64] Satya Katragadda, Paul J Darby, Andrew Roche, and Raju Gottumukkala. Detecting low-rate replay-based injection attacks on in-vehicle networks. *IEEE Access*, 8:54979–54993, 2020.

[65] Hyunsung Lee, Seong Hoon Jeong, and Huy Kang Kim. Otids: A novel intrusion detection system for in-vehicle network by using remote frame. In *2017 15th Annual Conference on Privacy, Security and Trust (PST)*, pages 57–5709. IEEE, 2017.

VITA

Ayan Roy was born in a city named Howrah, located in the state of West Bengal, India. Prior to his doctorate, he received his Bachelor of Science (BSc.) followed by Master of Science (MSc.) in Computer Science from St. Xavier's College, Kolkata, India, in June 2014 and June 2016 respectively.

Ayan received his Doctor of Philosophy in Computer Science from Missouri University of Science and Technology in July 2022 under the supervision of Dr. Sanjay Madria. His main area of research was analyzing the security and privacy issues in Vehicular Adhoc networks. More precisely, the primary goal of his research was to design and develop resilient algorithms that could foster a secured connected vehicle environment by removing malicious information and vehicles from the connected network, as well as developing algorithms for securing the in-vehicle network from remote attacks. During his Ph.D, he published several papers in top-tier conferences. He was also appointed as a Security Research intern at Samsung Research America in Summer 2021.

Besides academic involvements, Ayan enjoyed cooking, swimming, watching Soccer, going out on long drives, as well as playing online games.