A Byzantine-Tolerant Distributed Consensus Algorithm for Connected Vehicles Using Proof-of-Eligibility

Huiye Liu
huiyeliu@gatech.edu
Georgia Institute of Technology

Chung-Wei Lin
cwlin@csie.ntu.edu.tw
National Taiwan University

Eunsuk Kang
eskang@cmu.edu
Carnegie Mellon University

Shinichi Shiraishi
shinichi.shiraishi@tri-ad.global
Toyota Research Institute - Advanced Development, Inc.

Douglas M. Blough
doug.blough@ece.gatech.edu
Georgia Institute of Technology

ABSTRACT

Emerging applications in connected vehicles have tremendous potential for advances in safety, navigation, traffic management and fuel efficiency, while also posing new security challenges such as false information attacks. This paper targets the problem of securing critical information that is disseminated among nearby vehicles for safety and traffic efficiency purposes through distributed consensus. We present a consensus algorithm, which uses a "proof of eligibility" test to establish that a group of vehicles are actually within the vicinity of the information source. With the presence of a limited number of compromised (Byzantine faulty) participants, our algorithm provides correct consensus among healthy vehicles in real time. The algorithm provides fast and reliable consensus group formation and private key distribution without privileged members, trusted setup, or leader election. In addition to proving a safety property of our consensus algorithm, we have implemented it on top of a widely-used vehicle simulation environment (SUMO, OMNeT++ and Veins) and evaluated its performance on a model of the streets in a real midtown area. Simulation results demonstrate that the algorithm can reach consensus very efficiently (within 9.5s) and with up to 30% of compromised vehicles in a given area. The simulations also demonstrate the ability of our algorithm to more quickly disseminate information about a traffic accident and more efficiently route traffic around the accident site, as compared to previous robust information dissemination approaches.

CCS CONCEPTS

• **Networks** → **Application layer protocols**; • **Computing methodologies** → **Distributed algorithms**; • **Security and privacy** → *Mobile and wireless security*.

KEYWORDS

connected vehicles, distributed system, security

MSWiM '19, November 25–29, 2019, Miami Beach, FL, USA

ACM ISBN 978-1-4503-6904-6/19/11…$15.00
https://doi.org/10.1145/3345768.3355910

ACM Reference Format:
Huiye Liu, Chung-Wei Lin, Eunsuk Kang, Shinichi Shiraishi, and Douglas M. Blough. 2019. A Byzantine-Tolerant Distributed Consensus Algorithm for Connected Vehicles Using Proof-of-Eligibility. In *22nd Int'l ACM Conference on Modeling, Analysis and Simulation of Wireless and Mobile Systems (MSWiM '19), November 25–29, 2019, Miami Beach, FL, USA.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3345768.3355910

1 INTRODUCTION

Connected vehicle technologies are changing the ways we commute and communicate. The automotive industry is being completely reshaped by emerging communication technologies (V2V, V2X, V2I, etc.), mobility apps, electric vehicles, and vehicle automation. This provides an opportunity to reorganize our transportation infrastructure, improving safety, security, traffic efficiency, human comfort, and energy efficiency. For example, a connected vehicle can be informed if an emergency vehicle is approaching far away or a smart intersection can consider the estimated arrival times of connected vehicles to schedule those vehicles and increase the intersection throughput.

However, security becomes a paramount concern for connected vehicles, as they inevitably make control decisions based on information from other vehicles and external sources. In this scenario, the vehicular network is especially vulnerable to false information attacks. Fake messages can create problems like longer time required to reach destination, more gas consumption than needed, traffic jams, and even collisions. For instance, taking advantage of the connectivity, an attacker may compromise vehicles to lie about its own vehicle's position and speed, to make false warnings about a non-existent accident thus leaving an empty street for itself to freely pass through, or to report fake arriving time to a traffic signal control system causing congestion/traffic jams or even to ease criminals' getaways from crime scenes [28].

To date, research in automotive security has addressed many different perspectives. The goals include secure communication protocols integrated with existing standards and protocols at the external network layer, Intrusion Detection Systems (IDSs) and firewalls at the gateway layer, lightweight message authentication and encryption at the in-vehicular network layer, and Hardware Security Modules (HSMs), secure boot, and secret key control at the component layer.

Nevertheless, even with good security mechanisms at each of those layers, it is necessary to address security at the application

layer, especially for application information. In this paper, we address this issue through Byzantine-tolerant distributed consensus on application information. Several example use cases include:

Dynamic maps: weather condition, parking availability, EV charging station availability, etc. are all important information comprising a powerful dynamic map. Information reported by a single device/vehicle is not complete and reliable.

Traffic alerts: to provide trust in traffic condition warnings such as collision, congestion, and emergency vehicle(s) approaching, and to avoid inappropriate reactions, distributed consensus can be used to provide trusted alert dissemination.

Intersection management: the next-generation transportation system such as Intelligent Traffic Signal System (I-SIG) relies on vehicles' reported speed and location information to estimate the queuing line, and assigns green/red light time as needed. However, attacks [8] have been found to potentially cause serious problems.

Despite the conceptual appeal of this approach, realizing distributed consensus poses many challenges in connected vehicle systems. One challenge is *safety-critical operation* in the face of *real-time constraint*. For example, if an emergency vehicle needs to take priority at an intersection, this information is only important when it is approaching the intersection. After the vehicle passes, the information is no longer useful. However, forcing consensus within a short period of time could lead to incorrect decisions that may make the situation worse than without connected vehicle system. The second challenge is the *high mobility* and *communication loss/delay*, which could significantly affect the ability to reach consensus among devices. The third major challenge is the *lack of trust* in connected vehicle systems. Compromised vehicles with valid credentials will appear as trusted entities [1], which is a difficult situation to handle. Moreover, it is not necessary for compromised vehicles to always behave incorrectly. They may behave as healthy devices at one time and act incorrectly at another time, thus making it hard to rely on reputational trust. To address these challenges, we introduce a distributed consensus algorithm based on *Proof-of-Eligibility (PoE)*. To the best of our knowledge, this is the first work addressing distributed consensus in the face of the challenges listed above. Contributions of the paper include the following:

- We introduce the concept of Proof-of-Eligibility Challenge, which limits the impact of compromised vehicles from outside of an event area by preventing them from participating in the consensus process.
- We present the Byzantine-fault-tolerant consensus algorithm for connected vehicles (BFCV) to ensure information security among vehicles, without requiring privileged members, leader election, nor trusted shared key distribution. The algorithm also provides dynamic consensus group formation in an environment without a known pre-defined set of consensus participants.
- We report on the implementation of a BFCV prototype and simulation of it in a realistic environment built on top of Veins, SUMO and OMNet++. Evaluation results show that BFCV provides fast consensus satisfying both safety and liveness requirements.

The remainder of the paper is organized as follows. Section 2 discusses a motivating example, elaborating how the BFCV algorithm can be applied to a real-world problem. Problem formulation,

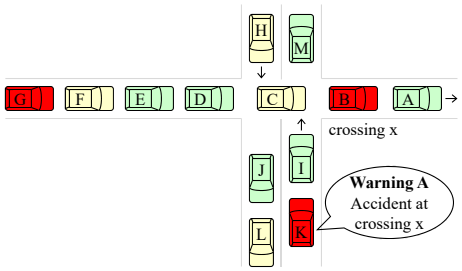

Figure 1: Example Scenario of a Fake Report

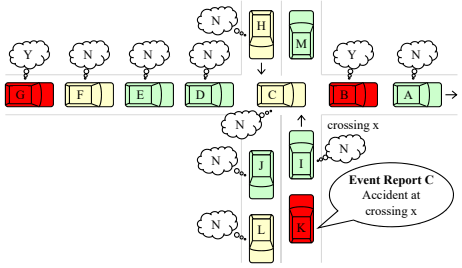

Figure 2: Example Scenario of a Fake Report with BFCV (where vehicles B, G, and K are compromised)

assumptions, threat model, and system model are provided in Section 3. We present the BFCV Algorithm in Section 4 and provide a proof sketch of its safety property. In Section 5, we present a concrete implementation of BFCV with details of environment set up, design of the experiments, and simulation results. Related work is discussed in Section 6 and we conclude in Section 7.

2 MOTIVATING EXAMPLE

In this section, we use dissemination of an accident alert as an example to illustrate how our proposed PoE-based consensus algorithm tackles the information security problem in connected vehicle systems. Figure 1 demonstrates a fake warning reported by compromised vehicle K. Each vehicle is labeled with letters as its name. Vehicles in green and yellow, representing different brands of vehicles, are honest nodes following the protocol and vehicles in red representing compromised nodes are trying to attack. Without a cooperative evaluation approach, a vehicle can only rely on its received data and local plausibility check as discussed in [15, 22, 31, 32] to make local decisions. Vehicles that are not able to "see" the crossing, may believe the false alert and could potentially reroute and transmit false information to further vehicles, if K is a compromised vehicle with valid credentials.

Our algorithm approaches this problem by using the concept of event reports, whose content could be an unconditional lane shift, slowing speed/congestion, observation of emergency vehicles, crashed vehicles, abnormal behaviors of neighboring vehicles, etc. In order for a created event report to be accepted by other vehicles in the network, a consensus group is formed with a group of eligible vehicles, who can solve a Proof-of-Eligibility (PoE) puzzle, to cooperatively evaluate the report content and reach consensus on whether or not the report is true. Only after that, the true report will be broadcast with group members' signatures. Upon receiving the

signed report, other vehicles will react accordingly after verifying the attached signatures.

As depicted in Figure 2, after vehicle K broadcasts an event report about an accident at crossing x, all vehicles within communication range (A-L) are able to receive the report. They try to solve a PoE puzzle attached in the event report to obtain a shared secret key. PoE is a set of consistency checks, which aims to prove that a vehicle is authentically relevant and eligible to participate in the cooperative evaluation of a reported event. The PoE puzzle is based on the local environment and can, therefore, only be solved by vehicles that are within close range of the event. PoE lessens the difficulties and shortens the delay of distributing shared keys among a temporarily formed group of moving vehicles. Let us consider a worst case scenario that both compromised vehicles B and G are able to solve the PoE puzzle and join the consensus group. During the consensus stage, compromised vehicles send false opinions agreeing with the fake report that there is an accident at crossing x while the honest vehicles dispute the report. Compromised vehicles B and G may also drop the consensus message received from other members to affect the group evaluation. However, as we will see later, with a minority of compromised vehicles participating, false consensus can never be reached with the BFCV Algorithm. In most cases, the group of honest vehicles will reach agreement that the report is false and disseminate a signed message repudiating the report.

In the above, the BFCV Algorithm description has been simplified to briefly introduce the general concept. A detailed algorithm description is provided in Section 4.

3 PROBLEM FORMULATION

3.1 Assumptions

In this paper, we assume active (engine-on) vehicles communicating with each other wirelessly. Vehicles routinely exchange information and monitor the environment, following these four steps: *Detection* – a vehicle detects new events (traffic condition, abnormal behavior, etc.) by receiving data from on-board sensors and surrounding vehicles. *Dissemination* – if a detected event is critical, a vehicle creates and broadcasts an event report to other nearby vehicles. *Decision* – upon receiving an event report, a vehicle evaluates the content of the report and makes a decision to accept it or not. *Reaction* – if an event report is accepted, a vehicle takes the corresponding action(s) such as to brake, accelerate, switch lanes, change routes, disseminate the event report, etc.

We mainly focus on dissemination and decision stages, which are the cornerstones of achieving reliable final reactions. Our goal is to identify potential security violations when attackers have the ability to tamper with information in messages and to mitigate the impact in a timely way. Before describing the threat and system models, we first state some assumptions: (1) The distributed system is asynchronous (unbounded communication delays) and we ensure the safety of our consensus protocol. However, liveness is not guaranteed unless enough messages are received within a time upper bound. (2) An attacker may exhibit compromised behavior at any point in time and remain benign at another time i.e., any vehicle in the network at time period $[t_i, t_{i+1}]$ can be compromised, even if it is behaving normally at time period $[0, t_i]$. (3) We assume that adversaries have limited computing power so that they cannot break the encryption and digital signatures. In other words, the cryptographic algorithms adopted are computationally secure. (4) Vehicles have public key certificates signed by trusted entities such as NHTSA [23] and/or vehicle manufacturers. (5) Private keys cannot be obtained by an attacker without a physical attack. However, by compromising a vehicle through software, an attacker can use an API to sign fake messages but does not know the actual key. This prevents remote attackers from stealing a valid private key from one vehicle and using it within a different vehicle or device.

3.2 System Model

We consider a set of vehicles that communicate by sending messages. We assume an unreliable communication medium where messages can be lost or delayed. A vehicle cannot receive other vehicles' messages if they are outside of the communication range (e.g., 200–300 meters for DSRC). Each vehicle can identify the sender of every message it receives by the sender's unique public key.

We assume that consensus begins with a vehicle generating an event report about conditions it observes on the road. The challenge, as expressed earlier, is for a set of vehicles nearby the event, that were previously unknown to each other, to form a group and reach consensus on whether the event report is accurate in a timely fashion despite the presence of compromised vehicles in the event area. Each vehicle that is nearby the reported event can form an opinion about whether the event report is accurate. We assume that non-compromised vehicles can correctly determine the accuracy of a report most of the time but occasionally a non-compromised vehicle might produce a wrong evaluation due to inaccurate or ambiguous sensing. We refer to vehicles that are not compromised but produce a wrong evaluation of an event report as *incorrect*. In this situation, it is useful for vehicles to learn a group opinion of the report accuracy to verify that their local sensor values are correct.

3.3 Threat and Fault Models

We are primarily concerned with attackers who compromise vehicles with valid credentials, and exploit improper/incomplete authorization checks. We adopt a very general threat model, where a compromised vehicle behaves arbitrarily (known as the Byzantine fault model), i.e. it may arbitrarily deviate from the protocol execution and can influence the data sent to communication channel. Through compromised software running on a vehicle, an attacker can broadcast any random or customized false data to the network, but cannot modify others' signed messages or otherwise interfere with others' message creation. In the most basic form, after successfully compromising a vehicle, typical exploits include withholding messages and sending out false data or irregular messages to others. Such attacks are successful when attackers can obtain compromised vehicles' valid certificates and credentials. Otherwise, the sent information cannot pass the authentication checks.

We assume that the number of compromised or incorrect vehicles within a small area is limited. To be specific, we assume that $f < u_{\min}/3$, where f is the number of vehicles in an area that are compromised or incorrect and $u_{\min}$ is a configurable parameter. Later, we will discuss how PoE puzzles can be used to limit participation in the consensus procedure to only those vehicles that are within the vicinity of the reported event area. This helps

to limit the number of compromised vehicles that can influence the consensus, allowing f and $u_{\min}$ to remain fairly small. This, in turn, improves the overall efficiency of the consensus operation and allows consensus to be completed faster, as compared to larger consensus groups that would be required for higher values of f. This also allows the total number of compromised vehicles in the network, beyond the vicinity of the reported event area, to be much larger than f.

Sybil attacks, first introduced in [11], are also a critical problem. An attacker launches a Sybil attack by creating multiple non-existent vehicles with valid identities spreading false information in the network. Various Sybil detection methods have been studied in the past, for example based on: directional antennas [29], received signal strength indicator (RSSI), fingerprinting [35], and interference-aware RSSI-based localization [12]. We assume that Sybil attacks are prevented by existing methods and so we do not consider them herein.

We also assume that there is no large-scale prearranged collusion between compromised vehicles to share answers to PoE puzzles. So, for example, a compromised vehicle does not predetermine PoE puzzles and distribute the answers to large numbers of other compromised vehicles. However, within a consensus group, compromised vehicles can collude arbitrarily (the Byzantine fault model, including collusion, applies within a consensus group).

3.4 Consensus Properties

Let Π represent a set of vehicles running a consensus algorithm on some event report R and let v_i be some vehicle in Π. Let x denote the correct evaluation result of R and $\bar{x}$ denote the opposite (incorrect) evaluation result. Finally, let $S_{ix} = \{v_j : v_i \text{ has heard } x \text{ from } v_j\}$, let $S_{i\bar{x}} = \{v_j : v_i \text{ has heard } \bar{x} \text{ from } v_j\}$, and let $S_i = S_{ix} \cup S_{i\bar{x}}$. S_{ix} contains the vehicles from which v_i has an evaluation result of x, $S_{i\bar{x}}$ contains the vehicles from which v_i has an evaluation result of $\bar{x}$, and S_i contains the vehicles that v_i knows about.

(1) *False Consensus* occurs when the following condition holds:

FC: $\exists Q \subseteq \Pi$ such that $|Q| > \frac{2u_{\min}}{3}$ and $\forall v_i \in Q, |S_{i\bar{x}}| > \frac{2|S_i|}{3}$ and $|S_i| \geq u_{\min}$ and $\forall v_i, v_j \in Q, i \neq j, S_i = S_j$.

Condition FC occurs when there is a group of vehicles of size greater than $\frac{2u_{\min}}{3}$ that all have heard the wrong evaluation result from more than 2/3 of the vehicles they have heard from and that also agree on the group membership at the end of algorithm execution. If this situation occurs with our BFCV algorithm presented later, a false event report will be disseminated. It is important to prevent this outcome.

(2) *Correct Consensus* occurs when Condition FC does not hold *and* the following condition holds:

CC: $\exists Q \subseteq \Pi$ such that $|Q| > \frac{2u_{\min}}{3}$ and $\forall v_i \in Q, |S_{ix}| > \frac{2|S_i|}{3}$ and $|S_i| \geq u_{\min}$ and $\forall v_i, v_j \in Q, i \neq j, S_i = S_j$.

Condition CC occurs when Condition FC does not occur and there is a group of vehicles of size greater than $\frac{2u_{\min}}{3}$ that all have heard the wrong evaluation result from more than 2/3 of the vehicles they have heard from and that also agree on the group membership at the end of algorithm execution. Note that if there are two large enough groups formed where one group agrees on the incorrect result and the other group agrees on the correct result, we still consider this to be false consensus. So, correct consensus is a large enough group agreeing on the correct result and the membership while no other large enough group agrees on the incorrect result. This is the ideal outcome for a protocol.

(3) *No Consensus* occurs when $\nexists Q \subseteq \Pi$, satisfying Condition CC or Condition FC.

This describes the situation where there is no consensus reached on either the evaluation result or the group membership or both at the end of algorithm execution. This situation would apply to algorithms that need a result within a certain time bound and terminate algorithm execution if it takes too long without reaching consensus. No consensus is preferable to false consensus but is still an outcome we would like to minimize. If the rate of no consensus is too high, valid event reports will not reach vehicles in a timely way, and this could potentially cause systemic problems.

4 BFCV ALGORITHM

4.1 Design Overview

BFCV has three features that differ from existing consensus algorithms that are targeted at connected vehicle systems.

First, it provides fast, reliable consensus group formation and shared key distribution without privileged members. The algorithm does not require trusted set up or leader election and only relies on very basic cryptographic assumptions. Each vehicle running on streets is considered as an untrusted entity equipped with valid credentials and some shares of knowledge describing the environment (traffic, weather, pedestrian, road-signs, etc.). Based on the assumption that at run time, the system does not know which entity is trustworthy and which is not, we do not follow the leader-election paradigm to construct evaluation groups. Instead, we use a set of challenge problems to perform plausibility checks, only allowing entities with sufficient proof of related knowledge and presence nearby the event location to join the cooperative evaluation.

Second, in most cases, BFCV guarantees all participating healthy vehicles reach agreement on the information being disseminated. In cases where the number of healthy vehicles is small or there is a very high rate of message loss, no consensus will be reached but false consensus will not occur (see Section 4.6 for a proof sketch).

Third, BFCV is agnostic to wireless communication technology as long as it supports inter-vehicle communication and underlying applications. No additional functionality such as remote cloud for computing, secure channel for message exchange, or trusted entities is required – it is fully distributed and self-maintained.

We next present the BFCV algorithm, which is divided into four phases: Report Generation, Proof-of-Eligibility, Evaluation Group Consensus and Report Verification. Notations used in the following sections are defined in Table 1.

4.2 Event Report Generation

An event report $R_E = (RID, E, EType, Cert_i, Q, \mathcal{H}(A), t_R, T_q)$, is generated and broadcast when a vehicle detects an unreported new

Table 1: Notation Table

v_i	Vehicle i
K_i^+, K_i^-	Public and private key of v_i
$Cert_i$	Certificate of v_i
D_i	Perception data of v_i
S_i	Hash table that records consensus status of v_i for different types of event reports
P_i	Event look-up table of v_i
F_i	Set of PoE challenge problem functions of v_i
E	An event
EID	An event's ID
$EType$	An event's type (collision, congestion, etc.)
R	An event report
RID	An event report's ID
Q	PoE challenge problems
A	Event Reporters' Answers to Q
$\mathcal{H}(A)$	Hash of A
t_R	Time stamp of report creation time
X	Signature
T_c	Time bound for reaching consensus
T_q	Time bound for solving Q, $T_q < T_c$
u_{min}	Minimum consensus group size
u_i	Membership list of v_i

event, where RID is the report ID, E is the event content, including location and estimated event life time based on criticalness, $EType$ is the event type, $Cert_i$ is the vehicle's certificate, Q is the PoE challenge problem, $\mathcal{H}(A)$ is the hash of computed answers to Q generated by the reporting vehicle, t_R is the event report time, and T_q is the time bound allowed for solving puzzle Q.

In real life scenarios, it is rare that within 200-300 meters, multiple critical events of the same type (such as rear-end collision, vehicle roll-over, etc.) exist. To improve the system efficiency and discourage compromised vehicles from flooding the network with fake reports, we allow only one event report for one $EType$ of event to be created and broadcast at a time. Once a report is broadcast, the reporter can neither join a different consensus group of the same event type, nor create another event report of the same type until the time bound of the consensus protocol for its current report is reached. If a vehicle does not follow the protocol and broadcasts a new event report before finishing the consensus time period, the inconsistency can be easily caught by other honest vehicles when they are solving the PoE challenge.

4.3 Proof-of-Eligibility

Before an event report can be accepted by other vehicles, it needs a valid group of vehicles to approve it. How vehicles are selected to form a valid evaluation group is the key to our proposed algorithm. We introduce the the concept of Proof-of-Eligibility to address this.

PoE challenge is powerful, but very application specific. In general, there is a large challenge problem pool pre-installed in vehicles. Let Φ denote the pool stored in vehicles, Q denote a selected challenge problem set from Φ, and function f denote a single problem in Q. Every time an event report is created, a Q will be automatically selected from Φ based on the event type, the event reporter's sensor data, the event time, and a randomly generated nonce. Each set Q must consist of problems of the following types:

View - this type of problem proves the vehicle's proximity to the target position and whether it has a potential view of the event. For instance, even if two vehicles are geographically close in distance, the two vehicles might be on two sides of a big building. Then if an accident happens on one side A, the vehicle on the other side is very unlikely to detect it. Thus, we consider that the vehicle on the other side has a close enough position but does not have a qualified view. Problems in this category use features such as position, speed, acceleration, speed limit, moving direction, colors of a nearby building, number of stop signs, etc.

Knowledge - this type of problem proves whether the vehicle has a certain amount of knowledge about the event of interest. For example, if the vehicle itself is at street S_a and the event is about whether the green vehicle moving on street S_a is compromised. If the vehicle does not even observe a green vehicle on S_a, it definitely has no knowledge of the event. Problems in this category include true or false questions such as whether the vehicle received a BSM (basic safety message) from location A or whether the vehicle's current speed is below 60 mph.

Consistency - this proves the consistency of a vehicle. Even if a vehicle gets all problems from category 1 and category 2 correct, its answers could have been lucky guesses. This type of problem aims to ask a sequence of true or false questions to further check whether the answers have inconsistencies. For example, questions in category 1 may ask the color of a building and a moving direction. Then question in this category may ask whether the vehicle can see another building of another color. However, the vehicle cannot see this color unless it moves in the opposite direction. If the vehicle's answer is true, then it fails the consistency test.

Algorithm 1: Proof-of-Eligibility Challenge

```
1 vehicle v_j receives R_E = (RID, E, Cert_i, Q, H(A), t_R, X) from v_i;
2 while (v_j is operating) ∧ (s_j[EType] = idle) do
3     if (t − t_R) < T_q and X is correct then
4         A' = RID;
5         for f ∈ Q do A' = concatenateBits(A', f(D_j)) ;
6         if (H(A') = H(A)) ∧ (t < T_q) then
7             obtain shared key K_R^- = KeyGen(A');
8             set S_j[EType] to busy until (t − t_R) > T_c or consensus
                is reached;
9     else drop R_E;
```

In the proposed algorithm, once an event report is received by a vehicle, it tries to solve the puzzle Q within time bound T_q; in the end, qualified vehicles are able to obtain a seed to feed into their local key generation function thus obtaining a shared secret key K_R^-. Vehicles then use the obtained K_R^- to initiate a $Hello$ message to other group members. By using the PoE puzzle, evaluation groups are able to form at run time without a selected leader, which saves the time spent on leader election and avoids the risk of granting privileges to a compromised leader. In our proposed model, all group members have the same privilege. The procedure of proof-of-eligibility is presented in Algorithm 1.

In practice, not all of the above problem types can be easily implemented, due to limitations of current vehicles. In our prototype,

we only implemented category 1 and 2 problems, excluding problems that require a camera and image processing. In our prototype, only problems that can be answered from existing sensors such as speed, acceleration, GPS location, etc., are implemented. Better PoE challenge design is the subject of future research. Emerging technologies will likely help with this. For example some newly emerged light flashing techniques [33] produce light not visible to drivers but allow vehicles equipped with cameras to capture it, and these techniques will work very well for PoE applications.

4.4 Evaluation Group Consensus

Reaching consensus in dynamic vehicular networks in a timely fashion is the key feature of our proposed algorithm. As described in the last subsection, a vehicle that successfully solves the PoE challenge broadcasts an encrypted hello message, M_H, with below format, to establish connections with other members:

$$M_H = Enc(K_R^-, Cert_i, R_E, x_i, Sign(K_i^-, R_E, x_i), \quad (1)$$

where x_i is v_i's local opinion of the event report value.

Once some connections among group members are established, encrypted consensus messages, M_C, with below format are sent to initiate voting consensus among members:

$$M_C = Enc(K_R^-, Cert_i, R_E, u_i, x_i, Sign(K_i^-, R_E, x_i), Sign(K_i^-, u_i), \quad (2)$$

where u_i is the known-member list by v_i. Each member in u_i is represented by its public key. The signature of $Sign(K_i^-, x_i)$ denotes attesting of the opinion by the sender, and $Sign(K_i^-, u_i)$ denotes attesting of the sender's recognized group member list.

Traditional consensus algorithms require fixed and known membership. However this is extremely hard to obtain in a highly dynamic vehicular network. We perform consensus on the group membership list G and the opinion list O simultaneously, instead of first agreeing on group membership and then initiating opinion consensus among agreed-upon members. Each element in G and O is uniquely linked to a group member which had its hello message received. For example, if a vehicle v_k receives a hello message from vehicle v_j, v_j is added to v_k's membership list, then $G_k[j]$ is initialized to 0 indicating that v_j is now a known member of v_k. O_k gets updated with $O_k[j] = 1$ if v_j agrees with v_k, otherwise, $O_k[j] = 0$. Moreover, $G_k[j]$ is set to 1 if the membership list of v_j is the same as that of v_k. This is realized by comparing the membership list obtained from M_H with v_k's local membership list. A consensus is reached when more than 2/3 of the vehicles in one vehicle's membership list agree both on the membership and the report value (opinion). The more than 2/3 requirement satisfies the well-known bound for Byzantine agreement. At this point, if the group membership size is at least as large as the minimum group size, then a decision message is broadcast to the network.

A time bound for consensus T_c is set to ensure the effectiveness of the algorithm in vehicular networks. If consensus is reached before T_c, the consensus process terminates with a decision message being broadcast. If consensus is not reached before T_c, the consensus process terminates and the related event report is dropped.

Additionally, we introduce a time variable t_{step} such that for every t_{step}, the vehicle broadcasts a message M_C even when there are no consensus messages or hello messages received. This is important when the vehicle's previously sent messages suffer from packet loss and the vehicle becomes disconnected from the other members. The detailed procedure is presented in Algorithm 2.

Finally, in order to tolerate packet loss, delay, and Byzantine behavior, we allow a vehicle to add another vehicle to its group list when it observes the vehicle in enough other vehicles' group lists even if it did not receive a hello message from the vehicle. The threshold we set for this is more than $\frac{1}{3}$ of the minimum group size to ensure that at least one healthy vehicle has heard a hello message from the new vehicle. In order to keep the code description fairly simple, we do not show this aspect in the pseudocode.

Algorithm 2: Evaluation Group Consensus

v_k obtained K_R^- by solving PoE challenges;
v_k receives a message M from v_j ($j \neq k$), set $t' = t$;
while $(t - t_R) < T_c \wedge$ *(consensus is not reached)* **do**
  if $t \geq (t' + t_{\text{step}})$ **then** set $t' = t$, create and broadcast M_C;
  if *M can be decoded using K_R^- and verified* **then**
    if *M is a hello message* $\wedge\ v_j \notin u_k \wedge\ !\Pi_{sync}$ **then**
      add v_j to u_k;
      if $x_j \neq x_k$, then $O_k[j] = 0$, otherwise $O_k[j] = 1$;
      $t' = t$, create and broadcast M_C;
    else if *M is a consensus message* $\wedge\ v_j \in u_k$ **then**
      if $!\,\Pi_{sync}$ **then**
        if $v_k \notin u_j$ **then** send hello message M_H;
        if $x_j \neq x_k$ **then** $O_k[j] = 0$;
        else $O_k[j] = 1$;
        if $u_j \neq u_k$ **then** $G_k[j] = 0$;
        else $G_k[j] = 1$;
        if $|\{l \mid G_k[l] = 1, O_k[l] = 1, l \in u_k\}| > \frac{2|u_k|}{3}$ $\wedge\ |u_k| \geq u_{\min} \wedge (t - t_R) > T_q$ **then**
          Π_{sync} = true and set consensus flag to true;
          create and broadcast decision message;
        else Π_{sync} = false;

4.5 Event Report Verification

A decision message is created if more than 2/3 of the vehicles agree on the same value and on the group membership. A decision message is denoted by:

$$M_D = (R_E, \alpha, CERTs, SIGs), \quad (3)$$

where α is the decision result, $CERTs$ is a set of certificates of the group members, and $SIGs$ is a set of signed opinions of the members in u such that for each v_i, $Sig_i = Sign(K_i, x_i)$.

When a vehicle receives a decision message and either it is not part of the consensus group or it is part of the consensus group but has not yet reached a decision, it examines the attached signatures. If the group size is at least $u_{\min}$, all signatures are valid, and more than 2/3 of the signed values agree with the decision value, the vehicle accepts the decision. In this way, vehicles that are not compromised but have the incorrect value will accept the group decision about the event's status. Any message with one or more invalid signatures or a group size less than $u_{\min}$ will be discarded. If multiple different decision messages regarding the same event report are received by a vehicle, it accepts the valid decision message with the longest signature chain and rejects the others.

4.6 Proof Sketch of Protocol Correctness

As is typical for distributed consensus protocols in challenging environments, the PoE protocol guarantees safety but not liveness. However, liveness is demonstrated through our simulation experiments described in Section 5.

Our main safety property is that false consensus does not occur as long as less than 1/3 of the vehicles in the vicinity of the event are compromised or incorrect. Thus, the only possible outcomes of the protocol are correct consensus and no consensus. This is detailed in the following claim and proof sketch.

Claim: Let the minimum consensus group size be $u_{\min}$. As long as the number of compromised vehicles and incorrect vehicles in the area of an event report $R_E(t)$ between the time of the report t and the time $t + T_c$ is less than $u_{\min}/3$, then false consensus cannot occur.

Proof Sketch:

The proof of eligibility challenge plays a fundamental role in ensuring safety. Only vehicles that can observe the area of the event report are capable of passing the challenge. This prevents compromised vehicles from *outside* of the event area from participating in the consensus. Thus, only compromised vehicles within the area of the report during the time that the consensus protocol is executed need be considered.

Additionally, we assume that the number of healthy vehicles in the event area that do not correctly verify the status of an event report is very small so that the total of compromised and incorrect vehicles in the vicinity of the report is less than $u_{\min}/3$.

False consensus requires agreement on the wrong value of an event report (e.g. "no accident" when an accident has actually occurred) and agreement on group membership. This could possibly occur in two situations: 1) when a vehicle correctly reports an event but enough compromised and incorrect vehicles within the formed consensus group conclude the event did not occur, or 2) when a compromised or incorrect vehicle falsely reports an event and enough other vehicles support the false report during the consensus procedure.

In either situation, there are two possibilities; either a consensus group of size at least $u_{\min}$ is formed for the event report or no large enough group is formed. If no large enough group is formed before time $t + T_C$, then no healthy node can broadcast a decision (see Lines 29–32 of Alg. 2 pseudocode) and the event report is dropped (this is a no consensus outcome).

If a large enough consensus group is formed, this means there are fewer than $u_{\min}/3$ compromised or incorrect nodes within the group that support the wrong event status. Thus, there are simply not enough nodes to broadcast the false evaluation value for any healthy node to accept it, since that would require more than $2u_{\min}/3$ false evaluations to be broadcast by distinct nodes. In this situation. If enough nodes that receive more than $2u_{\min}/3$ correct evaluation results also agree on membership of the consensus group, the result is correct consensus but if there are not enough nodes that agree on the membership, the result is no consensus. However, in neither case, can false consensus occur.

Figure 3: Evaluation Scenario - Urban

5 EVALUATION

5.1 Implementation

We implemented a prototype of BFCV in C++, which can simulate different scenarios by changing the map and system parameters. It is build on top of Veins [30] which provides a comprehensive suite of models of IEEE 802.11p, IEEE 1609.4 DSRC/WAVE and obstacle shadowing. We add additional layers to simulate packet loss/delay, cypto schemes supporting 128, 192, and 256 bit ECDSA keys for encryption, signing and verification, SHA-256 as the hash function. Our prototype consists of approximately 3500 lines of written code. Experimental maps are obtained from OpenStreetMap (OSM) [26]) with manual corrections of speed limit, traffic lights, number of lanes on the road, etc. to improve the accuracy. Vehicle mobility and routes are computed based on demand definition and shortest path algorithm using SUMO [17].

Different from previous works, the prototype includes realistic aspects of the vehicle dynamics (safe distance, mass, dimensions, vehicle types, braking distance, traffic lights, etc.), detailed modeling of the communication network, and real-life street maps with varying scales.

5.2 Simulation Results

Experiment Scenario: Our evaluation is conducted in a simulated midtown area of a major city in the U.S. with a capacity of around 700 moving vehicles (see Figure 3). We simulate a worst-case scenario where compromised vehicles behave honestly when there is no event to be reported. Thus, it is very hard for honest vehicles to catch bad behaviors prior to an event report. In the simulation scenario, a collision happens at a random time, and honest vehicles that detect the event create and broadcast "collision occurred" event reports leaving others to evaluate them. We also have compromised vehicles broadcast conflicting event reports saying "collision cleared" at the same time. Thus, there can be several consensus executions happening at the same time among different groups of vehicles to try to reach agreement to accept one of these conflicting reports. We also have compromised vehicles drop, delay or not send messages, and submit wrong opinions for evaluation.

BFCV Evaluation Results: Unless otherwise noted, the following parameters were used in all experiments: $T_q = 5s$, $T_c = 14s$, $u_{\min} = 7$, vehicle density = 250 and beacon message frequency = 10Hz. Natural packet loss and delay (not including compromised vehicles' behavior) were simulated such that messages were randomly dropped at receiving vehicles with a drop rate of 15% and

packets were randomly delayed within a range of 100ms - 1500ms. The communication range among vehicles was set to 300m based on NHTSA's proposed rule [24]. We use the following two metrics to evaluate the latency of BFCV:

- *Consensus Time:* the time spent on reaching consensus on a single event report. This starts from the event report creation time and ends when there is a decision message received by every member of the group contained in the message.
- *Decision Time:* the time spent on ultimately reaching consensus. This starts from the first event report creation time and ends when there is a decision message received by every member of the group contained in the consensus message. Note that this could involve multiple consensus attempts if the first attempt does not produce a consensus.

We first evaluated how BFCV's performance varies with minimum group size and vehicle density. We ran simulations with 10% of vehicles compromised, varied $u_{\min}$ from 4 to 10 with an increment of 1, and varied vehicle density from 50 to 450 vehicles with an increment of 100. 50 simulation runs were done for each parameter combination, where a single consensus period was simulated in each run. The possible results of each run are: Correct Consensus (CC), False Consensus (FC) and No Consensus (NC). Note that, as described above, there can be multiple consensus executions happening concurrently for "collision occurred" and "collision cleared" event reports. In case multiple large enough consensus groups succeed in reaching consensus, we record the result as FC as long as at least one of the groups agreed on "collision cleared". The results are shown in Figure 4.

Figure 4(a) shows the average consensus time in seconds vs. minimum group size and vehicle density. Note that NC outcomes are not included in the average, because there is no definite termination of the consensus in those cases. Not surprisingly, consensus time increases with both minimum group size and vehicle density since an increase in either parameter will cause the number of messages exchanged by the algorithm to increase. Figure 4(b)(c)(d) shows the different consensus outcomes vs. the two parameters. Note that, if the minimum group size is too small, compromised vehicles can form a group and reach false consensus. Also, if the vehicle density is too low, there are not enough vehicles in the event area to form a consensus group, and this leads to a high rate of NC outcomes. However, for a fairly wide range of group sizes and vehicle densities, there are zero FC results and a very low rate of NC outcomes. These results demonstrate that the choice of $u_{\min}$ should be based on both adversarial assumptions and expected vehicle density.

We also evaluated BFCV's performance versus the percentage of compromised vehicles with $u_{\min} = 7$ and a vehicle density of 250. For percentages from 5% to 40% with 5% increments, we repeated the simulation 50 times. The results are shown in Table 2. From the table, we can see that as the percentage of compromised vehicles increases, the percentage of CC decreases from 100% to 79%. However, even with 40% of the vehicles in the network being compromised, the BFCV Algorithm did not experience a single false consensus outcome.

We also evaluated how well BFCV handles failure to reach consensus (NC outcomes). Instead of stopping the simulation immediately after the single-round consensus timeout, T_c, occurred, we extended the simulation if consensus was not reached the first time. If a report evaluation fails to reach consensus within time T_c, then our algorithm drops the report. However, if this occurs in the simulation, another honest vehicle nearby will submit a new event report for consensus. By extending the simulation time, we examined whether the BFCV algorithm can recover from NC outcomes. In this case, we recorded the final decisions, i.e. whether there was ultimately a correct decision made after an event happened, possibly after more than one consensus attempt.

Table 2: Performance vs % of Compromised Vehicles

Mal_V	CC	FC	NC	AvgConsensusTime
5%	100%	0%	0%	5.228s
10%	99%	0%	1%	5.701s
15%	99%	0%	1%	6.206s
20%	99%	0%	1%	6.718s
25%	97%	0%	3%	8.542s
30%	93%	0%	7%	9.325s
35%	88%	0%	12%	10.446s
40%	79%	0%	21%	13.118s

Figure 5 shows the average decision time for different compromised vehicle percentages and different vehicle densities with $u_{\min} = 7$. When the percentage of compromised vehicles was low, the decision was made very quickly with an average that is well below the single-round consensus time bound T_c. However, as the percentage of compromised vehicles was increased, the time spent on evaluation rose. Note that, in some cases, the average decision time was close to or exceeded $T_c = 14s$, implying that more than one round of consensus was some times needed for those cases.

Comparison Results: We also simulated two related protocols, DC [27] and PoR [5], and evaluated them under the same experiment conditions. These protocols both use threshold-based voting, which is the most widely-used prior approach. Two metrics are introduced to compare the results:

- *Percentage of Vehicles Taking Action:* the percentage of vehicles in the network that reach a correct decision about the event report and take action to avoid the accident location
- *Average Commute Time:* the average simulation time that vehicles take to reach their destinations (vehicles not taking action to avoid the accident location experience a longer commute time due to backups around the accident site)

There was no explicit method described in [5] to set the threshold for the PoR algorithm. However, it should be based on the report criticality and the network status, which is similar to the minimum group size in our proposed algorithm. Therefore, we set both of these parameters to 7 in these simulations. The DC algorithm provides an explicit method for dynamically adjusting its threshold value, which we adhered to in our DC implementation. Other parameters of BFCV were the same as in the previous experiments.

Figure 6 depicts the percentage of vehicles taking action as the simulations progressed when 5% and 15% of vehicles were compromised, respectively. There are two main reasons why BFCV performed better than DC and PoR. First, with BFCV, vehicles make a group decision and act accordingly. For DC and PoR, each vehicle

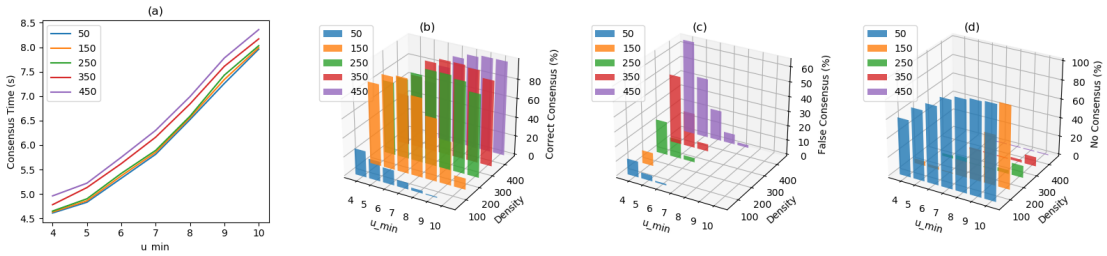

Figure 4: Consensus Time and Consensus Result vs. Minimum Group Size and Vehicle Density

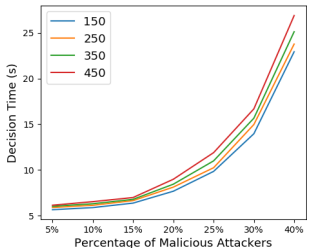

Figure 5: Decision Time vs % Compromised Vehicles

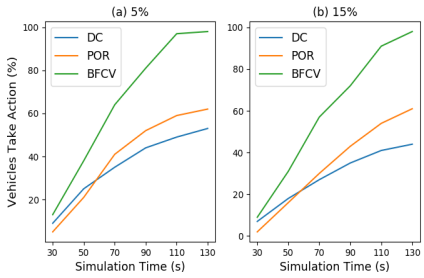

Figure 6: Vehicles Taking Action (%) vs Simulation Time

makes its own decision when enough endorsements from other vehicles are collected and, thus, not all vehicles make the same decision. In particular, since there are both "collision occurred" and "collision cleared" reports being circulated at the same time, some vehicles collect enough votes for "collision cleared" and reach the wrong decision even though most vehicles reach the correct decision. Second, BFCV uses PoE, which prevents compromised vehicles from outside the event area from participating. Since DC and PoR cannot verify location information of vehicles, they cannot filter out fake reports from compromised vehicles anywhere in the network that falsely report their location as being near the event, which increases the chances that enough votes can be collected to accept a fake report.

Table 3 depicts average commute time versus percentage of compromised vehicles ranging from 5% to 20%. For vehicles that were stuck in the simulation area at the end of the simulation, for the purposes of computing an average, we assigned them a commute time of 300 sec. From the table, we see that BFCV produced 15-22% lower commute times than PoR and 10-40% lower times than DC. As the percentage of compromised vehicles increased, most of the vehicles actually ended up being stuck for the DC Algorithm (avg. commute time approached 300 sec.), and a significant number were also stuck with PoR, while most of the vehicles actually reached their destinations during the simulated interval with BFCV.

Table 3: Avg. Commute Time vs. % of Compromised Vehicles

Mal_V	DC(s)	PoR(s)	BFCV(s)
5%	171	179	155
10%	194	197	162
15%	244	223	183
20%	286	248	203

6 RELATED WORK

In this section, we first review the existing approaches proposed for information security in connected vehicles. Node centric methods such as reputation systems [9, 10, 21, 31] inspect the past and present behavior of nodes and use this to predict the future misbehavior, which assumes that the nodes who behave well in the past are more likely to behave well in the future. However, smart adversaries may only initiate attacks at critical times, which is a fundamental problem that reputation systems cannot handle.

Data centric methods focus on analyzing transmitted data among nodes and information verification. Vehicles that use local methods, e.g. [13, 20, 31], verify information locally without relying on other vehicles' cooperation. Though these methods are light-weight, easy to scale and can tolerate intermittent communication, they heavily rely on location information and have a limited view of what is happening on the road, which reduces their accuracy. Cooperative schemes, as surveyed in [2], are more accurate than local methods with lower false positive and false negative rates. Nevertheless, they are more vulnerable to packet loss/delay and the ratio of compromised to honest vehicles. Threshold-based voting has been adopted in PoR [5], DC [27] to filter false data that honest vehicles only accept a report when they receive more than X signatures attesting to it. PoR improves the efficiency of communication by using growth codes, but the performance is sensitive to a preset threshold, while the DC method in [27] dynamically sets the threshold based on the criticality and the density of the network. However, [27] assumes 1-D communication, detection, and reaction, primarily limiting its use to highway scenarios. Moreover, threshold voting does not provide true consensus, because each vehicle decides independently and, therefore, different honest vehicles can reach different decisions.

True consensus algorithms satisfying *termination, agreement,* and *validity* properties have been well studied in other contexts. Paxos[18] and Raft [25] are well known algorithms to achieve consensus among unreliable nodes, but they do not address Byzantine faults [19]. Traditional Byzantine agreement protocols, e.g. [3, 6, 16] do not handle highly dynamic network connectivity such as occurs in vehicular networks. In general, traditional consensus algorithms require heavy computation, frequent message exchanges, and/or a fixed wired network, making them hard to be adapted to vehicular networks. While efforts have been made to adapt traditional consensus algorithms for dynamic and intermittent topologies in MANETs, e.g. [4, 7, 14, 34], none of these efforts address both unreliable links and Byzantine faults.

Although information security in vehicular networks has been previously studied, efficient collaborative methods that guarantee all healthy nodes make the same decision have not been developed to date. Our work presented herein provides an algorithm that provides timely and efficient consensus while supporting vehicles' high mobility as well as intermittent connections. In addition to achieving true consensus, our algorithm makes use of a novel proof of eligibility concept that prevents compromised vehicles from outside of an event report area from participating in decisions about the event. Prior approaches could not limit the participation of compromised vehicles in this way and were thus much more susceptible to intentional manipulation. As demonstrated in Section 5, our BFCV algorithm is the first to achieve true consensus with low latency in realistic vehicular scenarios with $\frac{1}{3}$ or more of the vehicles in the entire simulated area acting in a Byzantine faulty manner.

7 CONCLUSION

We presented BFCV, a distributed consensus algorithm based on "proof-of-eligibility", which achieves Byzantine agreement with unknown group membership and unreliable communication channels and is targeted at vehicular network environments. BFCV leverages the unique characteristics of moving vehicles and cryptographic primitives to prevent a large number of compromised and unrelated vehicles from joining the consensus group. This significantly speeds up the consensus procedure, providing a new paradigm of Byzantine-tolerant fast consensus for connected vehicles.

8 ACKNOWLEDGEMENT

This work is partially supported by Ministry of Education (MOE) in Taiwan under Grant Numbers NTU-107V0901 and NTU-108V0901 and Ministry of Science and Technology (MOST) in Taiwan under Grant Number MOST-108-2636-E-002-011.

REFERENCES

[1] M. Al-Kahtani. 2012. Survey on security attacks in Vehicular Ad hoc Networks (VANETs). In *Signal Processing and Communication Systems (ICSPCS), 2012 6th International Conference on*. IEEE.

[2] M. Arshad, Z. Ullah, N. Ahmad, M. Khalid, H. Criuckshank, and Y. Cao. 2018. A survey of local/cooperative-based malicious information detection techniques in VANETs. *EURASIP Journal on Wireless Communications and Networking* 1 (2018).

[3] J. Augustine, G. Pandurangan, and P. Robinson. 2013. Fast byzantine agreement in dynamic networks. In *Proceedings of the 2013 ACM symposium on Principles of distributed computing*. ACM.

[4] Abdulkader B., Pascale L., and Frédéric G. 2015. Solving Consensus in Opportunistic Networks. In *ICDCN*.

[5] Z. Cao, J. Kong, U. Lee, M. Gerla, and Z. Chen. 2008. Proof-of-relevance: Filtering false data via authentic consensus in vehicle ad-hoc networks. In *INFOCOM Workshops 2008, IEEE*. IEEE.

[6] M. Castro, B. Liskov, et al. 1999. Practical Byzantine fault tolerance. In *OSDI*, Vol. 99.

[7] D. Cavin, Y. Sasson, and A. Schiper. 2004. Consensus with unknown participants or fundamental self-organization. In *International Conference on Ad-Hoc Networks and Wireless*. Springer.

[8] Q. Chen, Y. Yin, Y. Feng, Z M. Mao, and H. Liu. [n. d.]. Exposing Congestion Attack on Emerging Connected Vehicle based Traffic Signal Control. ([n. d.]).

[9] Q. Ding, X. Li, M. Jiang, and X. Zhou. 2010. Reputation management in vehicular ad hoc networks. In *Int'l Conf. on Multimedia Technology*. IEEE.

[10] F. Dotzer, L. Fischer, and P. Magiera. 2005. Vars: A vehicle ad-hoc network reputation system. In *World of Wireless Mobile and Multimedia Networks, 2005. WoWMoM 2005. Sixth IEEE International Symposium on a*. IEEE.

[11] J. Douceur. 2002. The sybil attack. In *International workshop on peer-to-peer systems*. Springer.

[12] T. Garip, H. Kim, P. Reiher, and M. Gerla. 2017. INTERLOC: An interference-aware RSSI-based localization and Sybil attack detection mechanism for vehicular ad hoc networks. In *14th Annual Consumer Comm. & Networking Conf.* IEEE.

[13] M. Ghosh, A. Varghese, A. Gupta, A. A Kherani, and S. Muthaiah. 2010. Detecting misbehaviors in VANET with integrated root-cause analysis. *Ad Hoc Networks* 8, 7 (2010), 778–790.

[14] F. Greve and S. Tixeuil. 2007. Knowledge connectivity vs. synchrony requirements for fault-tolerant agreement in unknown networks. In *Dependable Systems and Networks, 2007. DSN'07. 37th Annual IEEE/IFIP International Conference on*. IEEE.

[15] J. Grover, Nitesh. Prajapati, V. Laxmi, and Manoj. Gaur. 2011. Machine learning approach for multiple misbehavior detection in VANET. In *International Conference on Advances in Computing and Communications*. Springer.

[16] R. Guerraoui, F. Huc, and A. Kermarrec. 2013. Highly dynamic distributed computing with byzantine failures. In *Proceedings of the 2013 ACM symposium on Principles of distributed computing*. ACM.

[17] D. Krajzewicz, J. Erdmann, M. Behrisch, and L. Bieker. 2012. Recent Development and Applications of SUMO - Simulation of Urban Mobility. *International Journal On Advances in Systems and Measurements* 5, 3&4 (December 2012).

[18] L. Lamport et al. 2001. Paxos made simple. *ACM Sigact News* 32, 4 (2001).

[19] L. Lamport, R. Shostak, and M. Pease. 1982. The Byzantine generals problem. *ACM Transactions on Programming Languages and Systems (TOPLAS)* 4, 3 (1982).

[20] T. Leinmüller, E. Schoch, F. Kargl, and C. Maihöfer. 2010. Decentralized position verification in geographic ad hoc routing. *Security and communication networks* 3, 4 (2010).

[21] Z. Li and C. Chigan. 2014. On joint privacy and reputation assurance for vehicular ad hoc networks. *IEEE Transactions on Mobile Computing* 13, 10 (2014).

[22] N. Lo and H. Tsai. 2007. Illusion attack on vanet applications-a message plausibility problem. In *Globecom Workshops, 2007 IEEE*. IEEE.

[23] NHTSA. [n. d.]. Federal Motor Vehicle Safety Standards; V2V Communications. https://www.federalregister.gov/documents/2017/01/12/2016-31059/federal-motor-vehicle-safety-standards-v2v-communications#h-1

[24] NHTSA. 2017. Federal Motor Vehicle Safety Standards; V2V Communications. Retrieved May 20, 2019 from https://www.federalregister.gov/documents/2017/01/12/2016-31059/federal-motor-vehicle-safety-standards-v2v-communications

[25] D. Ongaro and J. Ousterhout. 2014. In search of an understandable consensus algorithm.. In *USENIX Annual Technical Conference*.

[26] OpenStreetMap contributors. 2017. Planet dump retrieved from https://planet.osm.org . https://www.openstreetmap.org.

[27] J. Petit and Z. Mammeri. 2011. Dynamic consensus for secured vehicular ad hoc networks. In *Wireless and Mobile Computing, Networking and Communications (WiMob), 2011 IEEE 7th International Conference on*. IEEE.

[28] C. Qi and M. Z. Morley. [n. d.]. Connected cars can lie, posing a new threat to smart cities. https://theconversation.com/connected-cars-can-lie-posing-a-new-threat-to-smart-cities-95339

[29] K. Rabieh, M. Mahmoud, T. N Guo, and M. Younis. 2015. Cross-layer scheme for detecting large-scale colluding Sybil attack in VANETs. In *2015 IEEE International Conference on Communications (ICC)*. IEEE.

[30] C. Sommer, R. German, and F. Dressler. 2011. Bidirectionally Coupled Network and Road Traffic Simulation for Improved IVC Analysis. *IEEE Transactions on Mobile Computing* 10, 1 (January 2011). https://doi.org/10.1109/TMC.2010.133

[31] R. van der Heijden, S. Dietzel, and F. Kargl. 2013. Misbehavior detection in vehicular ad-hoc networks. *1st GI/ITG KuVS Fachgespräch Inter-Vehicle Communication. University of Innsbruck* (2013).

[32] R. van der Heijden, S. Dietzel, T. Leinmüller, and F. Kargl. 2016. Survey on misbehavior detection in cooperative intelligent transportation systems. *arXiv preprint arXiv:1610.06810* (2016).

[33] L. Wu and H. Tsai. 2013. Modeling vehicle-to-vehicle visible light communication link duration with empirical data. In *Globecom Workshops, 2013*. IEEE.

[34] W. Wu, J. Cao, and M. Raynal. 2008. Eventual clusterer: A modular approach to designing hierarchical consensus protocols in manets. *IEEE Transactions on Parallel & Distributed Systems* 6 (2008).

[35] Y. Yao, B. Xiao, G. Wu, X. Liu, Z. Yu, K. Zhang, and X. Zhou. 2019. Multi-channel based Sybil attack detection in vehicular ad hoc networks using RSSI. *IEEE Transactions on Mobile Computing* 18, 2 (2019).