Combining Detection and Verification for Secure Vehicular Cooperation Groups

MIKAEL ASPLUND, Linköping University, Sweden

Coordinated vehicles for intelligent traffic management are instances of cyber-physical systems with strict correctness requirements. A key building block for these systems is the ability to establish a group membership view that accurately captures the locations of all vehicles in a particular area of interest. In this paper we formally define view correctness in terms of soundness and completeness and establish theoretical bounds for the ability to verify view correctness. Moreover, we present an architecture for an online view detection and verification process that uses the information available locally to a vehicle. This architecture uses an SMT solver to automatically prove view correctness (if possible). We evaluate this architecture using both synthetic and trace-based scenarios and demonstrate that the ability to verify view correctness is on par with the ability to detect view violations.

CCS Concepts: • **Computer systems organization** → **Dependable and fault-tolerant systems and networks**; • **Security and privacy** → *Formal security models*; *Logic and verification*; • **Applied computing** → *Transportation*; • **Theory of computation** → Distributed algorithms;

Additional Key Words and Phrases: Vehicular coordination, formal verification, group membership, security

ACM Reference Format:
Mikael Asplund. 2019. Combining Detection and Verification for Secure Vehicular Cooperation Groups. *ACM Transactions on Cyber-Physical Systems* 1, 1, Article 1 (January 2019), 32 pages. https://doi.org/10.1145/3322129

1 INTRODUCTION

Rapid improvements in the development of cyber-physical systems in the transportation sector are on the verge of making self-driving vehicles ubiquitous in the society. Advanced driver assistance systems (ADAS) for road-based vehicles can already perform most of the low-level driving decisions, leaving only the high-level control to the driver. Vehicles are also becoming increasingly connected both through cellular technologies as well as inter-vehicle communication (IVC). Another strong trend is the emergence of delivery drones (aerial and ground-based), that might radically transform transportation infrastructures and enable a more sustainable way of living.

As these systems become prevalent, the need for reliable and efficient coordination mechanisms also increases. Several studies have shown the potential benefit of coordinated mobility in traffic management [9]. However, many coordination approaches today require pre-established groups, where the entities in the system have a-priori knowledge and trust of each other. The establishment of dynamic groups where the entities in a certain region form collaboration groups on-demand would provide significant advantages over the static approach. The concept of dynamic coordination groups can seen as a form of clustering [6] with explicit join and leave operations.

A fundamentally difficult challenge in providing dynamic group formation is to protect against faults and attacks. There are several reasons and methods for a malicious actor to attack a system

Author's address: Mikael Asplund, Linköping University, Department of Computer and Information Science, Sweden, mikael.asplund@liu.se.

XXXX-XXXX/2019/1-ART1
https://doi.org/10.1145/3322129

of vehicular entities. Potential incentives for attackers include extortion [18], theft of goods [13], selfish behaviour [22], automotive hacktivism [35], and possibly terrorism. There are also many non-malicious, but non-trivial faults that can have similar effects as attacks. Failing sensors, software bugs, and communication failure can all contribute to non-standard behaviour of a vehicle. Moreover, the possibility of combinations of accidental and malicious threats call for a joint threat model [25].

In this paper we study the problem of secure and dynamic group formation. In particular we focus on the ability of a vehicle in the group to ensure that the *group membership view* (i.e., the data structure containing information about the group members) that has been established is correct. To underline the importance of the group membership problem in vehicular coordination we first describe three problematic scenarios from the domain of intelligent transportation systems (ITS) and identify the possible causes (faults/attacks) that can lead to such undesirable effects. A group membership view can essentially be wrong in two ways. The first being that information about one or several members can be incorrect (e.g., by including a ghost member that does not exist in the physical world), which means that the view is *unsound*. The other way is if there are vehicles in the area that should be part of the view but have been omitted, in which case the view is *incomplete*.

We assume that each vehicle is equipped with some sensing capabilities to detect and identify other vehicles in its surroundings as well as IVC capabilities. Moreover, we consider the situation where a number of the vehicles experience some software or hardware fault that prevent them from behaving as proper group members. This model also covers transient faults such as packet loss, and problems with sensors. The question we study is whether the remaining non-faulty vehicles can correctly differentiate between correct (sound and complete) and incorrect (unsound and/or incomplete) group membership views. In particular, our hypothesis is that a vehicle should be able to *verify* that the view is correct based on the locally available knowledge.

As such, the concept that we investigate in this paper is a very general one. Given some violation detection mechanism, when do we know that a non-detection means that the current state is really a good state? If the detector is perfect (never makes a mistake), then non-detection is always the same as verification. But if there are cases where the detector fails to detect a violation (due to lack of knowledge), we need a separate mechanism for verification. That way we are able to differentiate between the case where a violation was not detected due to insufficient data and the case where a violation was not detected since we know that a violation would contradict the known data.

We formally define correctness criteria for membership views in a dynamic context and present the problem of violation detection and view verification for group membership using an abstract model of locations, vehicles and their sensing capabilities. Within this model, we establish two basic impossibility results for the ability to verify soundness and completeness respectively. Moreover, we propose an architecture for online detection and verification that can be deployed in a vehicle to analyse group membership views. This architecture uses a Satisfiability Modulo Theories (SMT) solver to automatically enumerate all the possible physical configurations that would match the known data. We evaluate our approach by testing a large number of synthetically generated and trace-based scenarios to understand under what circumstances view verification is feasible. The evaluation covers both fundamental properties of the graph models as well how to go from realistic road and vehicular mobility traces to a format that can be used by our verification framework. Finally, we consider consider how the framework is affected by staleness resulting from node mobility and lack of communication.

To the best of our knowledge, this is the first paper to provide a formal *on-line* model of view membership correctness and be able to verify view correctness for non-trivial scenarios. To summarise, the contributions of this paper are

- Formally stating the group verification problem for mobile entities

- Establishing theoretical impossibility results on the feasibility of verification
- A novel approach for secure group formation based on a formal model of the environment and the detection capabilities.
- Implementation and evaluation of the verification approach using an SMT solver.

The remainder of this paper is organised as follows. After the introduction, we will provide an overview of related work (Section 2), with a focus on security in vehicular networks and group management. Section 3 presents a number of vehicular coordination scenarios and relate them to the problem of group membership. In Section 4 we formally describe the system model and in Section 5 we prove some fundamental bounds on the feasibility of verification based on these concepts. Section 6 describe our approach to ensure dynamic group verification, which is evaluated in Section 7. Finally, Section 8, concludes the paper.

2 RELATED WORK

The security of vehicular communications has been studied from a range of different perspectives [32]. Given the legacy from research on MANETs many of the earlier works focused on security issues with regards to routing [21]. As the technology of inter-vehicle communication matured and became standardised, the focus shifted to the challenges of authentication, privacy [28], and how to prevent spreading of false information [7].

There are several works that propose security measures based on some kind of data-verification. Dietzel et al. [12] use clustering to filter out false information in an aggregation-based protocol for information on vehicle speed in an area. Jaeger et al. [17] use Kalman filters to predict mobility movements of surrounding vehicles and compare that with the actually received information. This allows the system to identify non-plausible vehicle movements. Generally, verification can only be done if there are two independent information sources that can be compared. Aslam et al. [2] explore mechanisms to forward data using independent groups of vehicles, either by separation in space or by separation in time.

Security in the context of vehicular platoons have been investigated by Studer et al. [34] who employ a combination of ensuring validity of data over time to verify that a vehicle is travelling in the same convoy, and distance-based verification using time-of-flight of messages and MAC-layer timestamps. Lyamin et al. [26] present an algorithm for detecting jamming of messages in a platoon. More recently Ucar et al [36] show that visible light communication can reduce but not remove the risks associated with attackers outside a platoon.

Cryptographic approaches will be essential in future vehicular communication solutions to guarantee privacy and integrity of data, as long as they are sufficiently lightweight [29]. Our work assumes basic cryptographic primitives (e.g., certificates and signed messages to prevent spoofing), but is otherwise orthogonal to approaches such as SPGS [19]. The work by Han et al. [16] provide an interesting alternative to the sensor-based approach in our work, where instead the accelerometer signals from a group of vehicles can be used to form a shared group key.

A key factor that separates our work from those mentioned above is the use of formal reasoning to support security and fault tolerance. There are previous works that use formal verification to prove such properties for distributed protocols. See for example Li et al. [23] and references therein. Ramasamy et al. [31] use the SPIN model checker to verify correctness of an intrusion-tolerant group membership protocol for static vehicle groups. In the vehicular context, Primiero et al. [30] present formal system to represent reputation in vehicular networks. Asplund et al. proved correctness of intersection collision avoidance using an SMT solver [5]. Common for these works is that the verification process is offline rather than online as in our current work.

The concept of group membership emerged as an elegant abstraction layer for tackling the consensus problem in distributed systems. While the node crash was the dominating fault model, there has been significant work on tackling network partition faults as well. Transis [1] was the first system that allowed partitions to continue with independent groups. For a comprehensive comparison of different group communication services, we recommend the paper by Chockler et al. [10]. More recently Lim and Conan [24] thoroughly address the problem of group membership in mobile ad-hoc networks. Fathollahnejad et al. [15] analyse the problem of group formation algorithm in vehicular networks. The authors make a distinction between safe and unsafe disagreement where unsafe disagreement occurs if vehicles decide on different non-empty views.

There is also a rich body of work on the problem of location verification [33, 38, 39]. In these works, it is typically assumed that some kind of range measurement exists (e.g., based on properties of the radio channel), and then a group of verifiers will try to determine whether a sending vehicle located where it claims to be. Our approach has similarities to these works but employs a more general perspective of the problem. First of all, we also consider the ability to detect hidden nodes. Moreover, the input to our verification procedure can be any kind of on-board sensors (where location verification can be one such input).

3 VEHICULAR COORDINATION SCENARIOS

The purpose of this section is to motivate the need of trustworthy membership views. We start this discussion by listing three well-known ITS applications where a notion of group membership is required. For each of these applications we then describe a scenario where the inability to have the correct view information could have negative effects on traffic efficiency and safety. Finally, we analyse and generalise these scenarios in terms of what faults and attacks that can give rise to incorrect membership views.

3.1 ITS applications that require group membership

We consider three commonly discussed future ITS applications. They differ in several ways, but have in common that they require coordination among a set of vehicles.

- Managed intersections. Virtual traffic lights have been proposed and investigated in a number of studies [9]. The gains compared to traditional traffic lights are obvious with increased throughput and reduced risk of collisions.
- Managed roundabouts. This form of managed intersection is a much less investigated topic. Roundabouts by themselves provide improved traffic flow compared to traffic-light operated intersection, but can be improved even further with the help of V2V communication [20].
- Platooning. Joint longitudinal control of a set of vehicles travelling along a line has been the topic of much research and was demonstrated as early as the 1990s in the PATH project. Today, this technology is becoming increasingly mature and ready for market deployment and the research focus is more on interoperability and added capabilities such as merging [27].

The requirements of these high-level applications are similar but not identical. The intersection application in its simplest form just replaces the coloured light sent out by the traffic light with radio messages exchanged between the vehicles. The actual movement of the vehicles would then be controlled as is done today (i.e., by a human driver or automated on-board driving system). The roundabout application requires more precise control of the vehicles to be of any added value. The vehicles must follow an agreed trajectory to create a smooth flow through the circulation place. The platooning application also requires a joint control scheme for the vehicles in the platoon.

In terms of the requirements on the membership layer, all three applications require a well-defined notion of which vehicles that are involved in the coordination. For the platooning case,

this membership view can be agreed on a long-term basis and possibly even statically or manually with the help of a cloud-based solution. The membership view follows the vehicles as they move along the road. Contrary to this, the intersection and roundabout applications require a much more dynamic notion of group views, were vehicles enter and leave the view as the enter and leave the area. In these cases a static solution or manual solution to group formation is not feasible.

3.2 Fault and attack scenarios

Based on these basic ITS applications we present three scenarios to illustrate the importance of correct membership views. The scenarios range from fairly benign to very dangerous.

Scenario 1. Hidden vehicle Figure 1 shows the first scenario based on a three-way intersection in an urban environment. The premise is that there is no centralised management of the intersection, but rather that the vehicles should coordinate themselves. An instance of this case was recently deployed in the 2016 GCDC event [14]. The figure shows how a vehicle A is about to enter a main road (going east-west), and must give way to the vehicles on this road. A building in the corner blocks the line of sight so that A cannot immediately see which vehicles are to its right. However, assuming that the vehicles are equipped with IVC capabilities, A might still learn about the other vehicles in the neighbourhood.

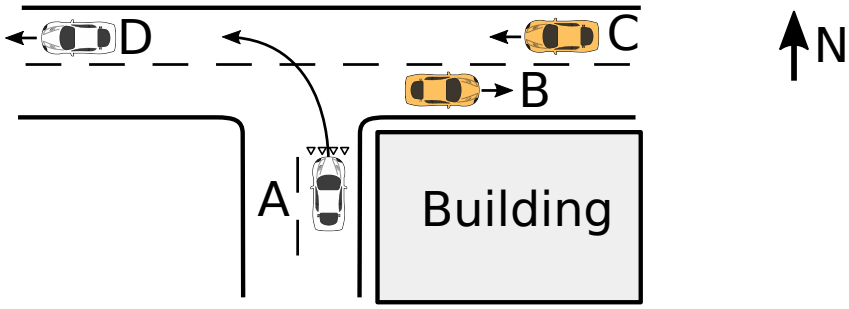


Fig. 1. Hidden vehicle scenario. Vehicle A wants to enter the main road and receives erroneous information from vehicle B that there are no hidden vehicles.

Now consider the case where vehicles B and C are faulty (as indicated with the orange colour). Vehicle is faulty C in the sense that it cannot send messages (possibly because it does not have IVC capabilities), and vehicle B in the sense that it fails to detect vehicle C using its sensors. In such a case A might falsely infer that only A, B and D are present and decide to enter the main road. Most likely, an on-board safety system would break before a collision occurs, but it would still be a potentially hazardous situation.

This scenario hinges on the fact that A uses the faulty sensors of B to determine the presence of hidden vehicles. If only on-board sensors are to be trusted, then A would have to be more careful in the crossing. On the other hand, this would limit the applicability of IVC technologies as an enabler for more efficient traffic flows.

Scenario 2. Selfish behaviour The second scenario is depicted in Figure 2. The scenario is based on single-lane roundabout with four entries and exits. As is usual in most countries, the vehicles that enter the roundabout must give way to the vehicles that have already entered. The consequence of this is that a vehicle that tries to enter when there is a continuous flow of vehicles in the roundabout must potentially wait for a long time. In the figure there is a flow of traffic from east to west and the vehicle A must wait.

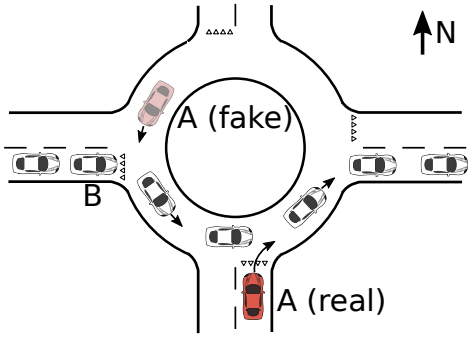


Fig. 2. Selfish behaviour scenario. The red vehicle A is trying to gain advantage by introducing a fake vehicle that would require vehicle B to wait and therefore allow A to enter the roundabout.

However, A can be selfish and either lie about its own position or invent a new virtual vehicle as indicated in the figure by a light red vehicle and denoted "A (fake)". The result is that vehicle B must wait for this invented vehicle, which will give A a chance to enter the roundabout, thereby cheating the system.

While this is a benign scenario, if this strategy is used by many vehicles this could result in unnecessary traffic congestion. It may be argued that vehicle B would immediately detect that the fake vehicle does not exist and is therefore able to ignore it. However, the fact that it receives the position and speed beacon from the fake vehicle using a V2V communication interface should at least result in a much more careful entry to the roundabout. This delay might be enough for A. From B's perspective the inconsistency between sensors and IVC information could also be explained by faulty sensors or other uncertainty factors.

Scenario 3. Platooning attack

The third scenario is based on an intentionally malicious attacker or malware. Figure 3 shows four vehicles travelling along a highway. Vehicle B is a regular vehicle without any IVC capabilities. Vehicle A is an attacker that has initiated a platoon with itself as a leader, but lying about its position to be in the location of B. Vehicles C and D join the platoon thinking that the leader vehicle is A.

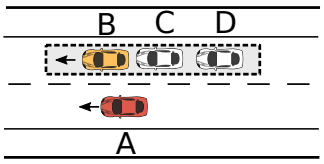


Fig. 3. Platooning attack scenario. Vehicle A pretends to be the leader of the platoon, as vehicle B is unaware of the situation.

If B suddenly has to break, and at the same time A sends out a signal to C and D that the leader has started to accelerate, there is a high collision risk, especially since vehicle D will take a long time to realise that the information provided by A was false. This scenario has been further analysed by Boeira et al. [8].

3.3 Fault/attack levels

We can identify a number of faults and attacks that could potentially give rise to these scenarios.

Communication failure In scenarios 1 and 3 above, some of the vehicles failed to communicate properly about their position and speed. This can be attributed to a number of factors including legacy vehicles without IVC capabilities, temporary failure communication components, and malicious behaviour.

Detection omission All three scenarios contain some kind of (possibly intentional) omission in the ability to detect surrounding vehicles (or the lack of them). In scenario 2, a vehicle fails to (immediately) detect that another vehicle is not present at the claimed position. In scenario 1, vehicles A and B fail to detect vehicle C, and in scenario 3, vehicle C fails to determine the identity of vehicle B.

Location falsification Scenarios 2 and 3 involve one vehicle actively lying about its position (or of an invented virtual vehicle). This requires malicious intent.

Sybil attack Finally, scenarios 2 and 3 could also be the result of a Sybil attack where a vehicle invents multiple identities.

We will return to these faults and attacks when specifying the fault model in Section 4.6. Note that there are several potential ways these attacks and faults can be detected and mitigated. However, for the purpose of this paper, we are mainly concerned with their impact on the correctness of group membership views. Note also that having a correct membership view would prevent the scenarios above, but the underlying faults described here can still cause harm to the system in other ways.

4 FORMALISING SECURE GROUP MEMBERSHIP

The core idea in this paper is to make use of intrusion detection capabilities in order to provide verification of views. In this section we provide an overview of a detection and verification framework, and then describe the basic system model we have used to capture the essential properties of this architecture. In the following sections we will then derive two basic theoretical results on the verifiability of a view under different model parameters (Section 5), and present an automated framework for online verification (Section 6).

Please note that the modelling choices made in this paper have been carefully made with regards to two often conflicting goals. Providing a high level of realism and expressiveness in the models increases the likelihood that they will be useful in real-world contexts. On the other hand, this also risks making them computationally intractable for automated theorem proving. Maintaining the balance between expressiveness and tractability is quite challenging. We believe that we have found an appropriate balance for the particular problem of verification of group membership views.

4.1 Overview

Figure 4 shows our architecture for violation detection and view verification. On the left side are the input sources that provide information to the system. This includes information coming from other (faulty and non-faulty) vehicles and the sensor information coming from on-board sensors such as radar, LIDAR, cameras, etc. The task of the *Group membership protocol* is to use this information to provide an as accurate view of the vehicles in the vicinity as possible. We have previously presented a protocol for this purpose [4].

The role of the *detector* is to check for any inconsistencies in the perceived information with regards to the group view. Examples of such inconsistencies can be that the sensor data does not match with information that is coming from the network. If the detector finds an inconsistency,

an alert is raised, otherwise not. There are numerous such detector mechanisms described in the literature (e.g., [26, 40]).

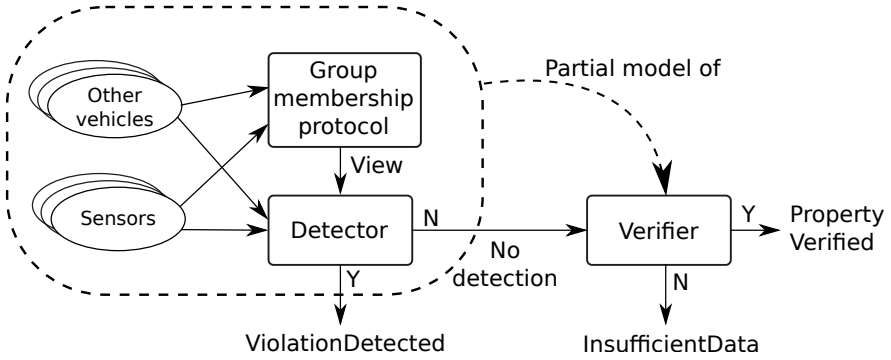


Fig. 4. Overview of an architecture for secure membership views

The key novelty in our proposal is the next part of the process. Rather than settling for non-detection of alerts we feed the data, together with a *model* of the system and the view to a verifier component that will assess whether a violation can be *ruled-out* or not. If the view can be verified to be correct (under reasonable assumptions), this significantly increases the ability to trust the information and thereby make safety-critical coordination decisions.

Intuitively, view verification works by considering all possible alternative explanations that would cause the same information as the vehicle currently sees. This typically includes both the most straightforward explanation (that things are as they appear), but also all the "what-ifs" such as what if there is a node just outside the sensing reach for which the communication has failed, would the available information then still be the same?

Note that the ability to detect violations and the ability to verify view correctness both depend on how well the vehicle that wants to do this is able sense its surroundings and communicate with non-faulty neighbours. If the vehicle is unable to sense its surrounding and is not able to rely on its neighbours (since they might be faulty), then neither detection nor verification will work. Due to the lack of information the view verification step will find a large set of possibilities that could potentially be true given what is currently known. The good part is that a lack of view verification is a reason for the vehicle to be extra careful (which seems to be a good idea in such cases).

In the following subsections we now describe the basic concepts we have used to model this architecture. The final subsection summarises the notation and provides a more intuitive explanation.

4.2 Vehicles and their locations

We model the system as a finite and non-empty set of locations L, and a discrete and non-empty set of vehicular nodes N (we will sometimes refer to a vehicular node simply as node). The physical configurations of vehicles at the locations is modelled using a physical configuration function $c : L \mapsto N \cup \{e\}$, where $e \notin N$ represents emptiness. We assume that no vehicle can be at two different locations at the same time.

We have opted for a discrete representation of location rather than a continuous one. We will return to the implications of this assumption and how to create a discrete representation from a continuous model of the environment in Section 6.3.

4.3 Sensor capabilities

We use a high-level abstraction of the ability to sense other nodes that is primarily based on the physical characteristics of the environment. This means that we assume all vehicles to have essentially the same sensing capabilities. Formally we model this as an undirected graph $G = (L, S)$ where L is the set of locations and the set of edges S represents the the ability to sense objects at one location from another. If a location l_i can be sensed from l_j (and vice versa) then $\{l_i, l_j\} \in S$. This model can be extended to incorporate asymmetric sensing capabilities, but we use an undirected graph for a more straightforward presentation.

For easier notation, we introduce a sensing function s that maps a vehicle to a set of locations that can be sensed from that vehicle. Given a sensing graph $G = (L, S)$ and a vehicular node set N, the sensing function $s : N \rightarrow 2^L$ is defined as $s(n_i) = \{l_j : \{l_i, l_j\} \in S \wedge c(l_i) = n_i\}$

4.4 Group membership views

We consider a view to be a representation of vehicles and their locations (i.e., it should in the best case capture the true state of the physical configuration).

Definition 1 (view). *A* **view** *is a non-empty set of node-location pairs* $v = \{\langle n_1, l_1\rangle \ldots \langle n_{|v|}, l_{|v|}\rangle\}$, *where vehicle nodes* $n_i \in N$ *as well as locations* $l_i \in L$ *are all distinct.*

For ease of notation we introduce N_v as a short-hand for the set of all vehicles in the view $N_v = \{n_1, \ldots, n_{|v|}\}$, and L_v for the set of all the locations in the view, $L_v = \{l_1, \ldots, l_{|v|}\}$. Note that to reduce notation cluttering we refrain from including unique view identifiers in the view as is common in literature on group membership. Such identifiers will be required in a real-world implementation to keep track of view evolution over time, but for the purpose of the presentation in this paper, they are not needed.

4.5 View properties

The only requirements for the views so far is that the vehicle nodes and locations are distinct. We now introduce two separate correctness properties that relate how well a view represents the real world (as modelled by the configuration function c). In previous work [4], we made similar non-formal definitions for view correctness as well as a notion of freshness. Here we remove the time-aspect and provide a stronger theoretical foundation to be able to formally prove view properties.

First, we define soundness of a view to mean that all the information contained in the view is correct (i.e., the vehicles are actually located where the view claims they are located). Note that view soundness is thus not just a property of the view itself, but rather a property of the view with respect to a configuration c.

Definition 2 (Soundness). *A view* v *is* **sound** *w.r.t. configuration* c *iff* $c(l_i) = n_i \neq e$ *for all* $\langle n_i, l_i\rangle \in v$.

View completeness requires that all vehicle nodes in the area (as described by the set L) are also part of the view. That is, the view should be complete with respect to all vehicle nodes in the area.

Definition 3 (Completeness). *A view* v *is* **complete** *w.r.t. configuration* c *iff for all locations* $l \in L$ *such that* $c(l) = n \in N$, *the corresponding node-location pair is in the view* $\langle n, l\rangle \in v$

A system that is capable of always providing sound and complete views will know about all the vehicles in the environment. If this was the case, then none of the fault scenarios from Section 3 could occur. A view which does not meet a particular criterion (soundness or completeness) with respect to a configuration is said to *violate* the criterion.

In distributed algorithms literature, properties on view membership is often phrased in terms of what one node perceive of other nodes. A node that in this sense believes in a view is said to have *installed* the view. The criteria we have defined above only consider the view itself in relation to the configuration. We do not consider whether vehicles have installed a view or not. While this is an interesting aspect, our goal is to separate the specification of views and the dissemination of these views.

4.6 Fault model and system specification

In Section 3 we presented three risk scenarios and listed a number of faults and attacks that alone or in some combination could result in those scenarios. Here we translate and specify those faults and attacks in terms of behaviour of the vehicular nodes in our model through the following assumptions:

- There are at most $f < |N|$ faulty vehicle nodes. We use $F \subset N$ to denote the set of faulty nodes $|F| = f$.
- A faulty vehicle may omit to send information, and may send false information.
- Multiple faulty vehicles may collude to send information that is consistent with each other but still false.
- Faulty vehicles are not able to corrupt or stop communication between non-faulty vehicles.
- Faulty vehicles are not able to overload the system by sending too many messages.
- Faulty vehicles will not try to warn other vehicles about violations against view soundness or completeness.

We model faulty nodes as black boxes, meaning that faulty behaviour is captured through what they send. The fact that faulty nodes can have faulty sensors is thus modelled by those nodes transmitting false information about what they have sensed. Apart from the send and receive omissions from faulty vehicles, we do not assume that channels can lose or corrupt messages. In reality, wireless channels in vehicular networks are quite lossy. However, over a longer time interval the likelihood of a message being successfully transmitted is still very high.

The final assumption in the list removes the ability of faulty vehicles to raise a false alarm, which would effectively prevent other nodes from forming a group. The rationale for this assumption is that given the high assurance level of automotive software malicious vehicles should be a rare occurrence (thereby making false alarms a minor issue). If, on the other hand, malicious vehicles will be common, then the whole idea of safety-critical vehicular coordination seems like a dangerous endeavour.

4.7 System Specification

We will now try to summarise the notation introduced in the previous subsections. We do this by first introducing the term *System specification* as a tuple $S = \langle N, G, f \rangle$, where N is the set of vehicle nodes, G is the sensing graph, and f is the maximum number of faulty vehicles. We assume that the system specification is static and that is is globally known by all nodes. Table 1 summarises all the notation introduced so far.

We now attempt to provide a more intuitive explanation of how to interpret the notation relating to the system specification (referring back to the relevant subsections). Figure 5 contains a very simple example that we use to illustrate the key concepts. We reuse variations of this example later in the paper as well. The set of vehicle nodes are shown in the lower part of the figure as rounded boxes. The set of vehicles is $N = \{n_1, n_2, n_3, n_4\}$ (see Section 4.2).

The upper part of the figure shows the sensing graph G (Section 4.3) which is composed of a set of locations $L = \{l_1, \ldots, l_8\}$ and the sensing edge set S that corresponds to whether two locations

Table 1. Notation summary

Vehicular node set	N
Locations	L
Emptiness	e
Physical configuration function	$c : L \mapsto N \cup \{e\}$
Sensing graph	$G = (L, S)$
Sensing function	$s : N \rightarrow 2^L$
View	$v = \{\langle n_1, l_1\rangle \ldots \langle n_{\lvert v\rvert}, l_{\lvert v\rvert}\rangle\}$
Vehicles in view v	$N_v = \{n_1, \ldots, n_{\lvert v\rvert}\}$
Locations in view v	$L_v = \{l_1, \ldots, l_{\lvert v\rvert}\}$
Maximum number of faulty vehicles	f
System specification	$\mathcal{S} = \langle N, G, f\rangle$

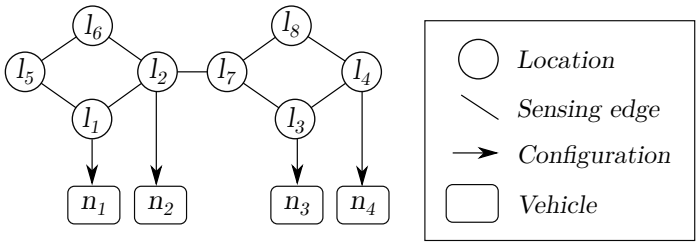


Fig. 5. Illustration of a simple system specification

are within sensing range of each other . In this example we see that for example l_1 and l_5 are within sensing range since they are neighbours in the graph, but l_5 and l_2 are not. The arrows going from the locations to the vehicles represent the configuration function (Section 4.2) c. The fact that vehicle n_1 is located in location l_1 is thus represented by $c(l_1) = n_1$. Locations for which there is no outgoing arrow are empty (e.g., $c(l_5) = e$). The sensing function s which is introduced in Section 4.3 to create a more shorthand notation captures all the locations that can be sensed by a particular node (under a particular configuration). So for node n_1 this would simply be $s(n_1) = \{l_1, l_2, l_5\}$.

Section 4.4 introduces views as a set of node-location pairs. In our example, the only view v which is both sound and complete (see 4.5) is $v = \{\langle n_1, l_1\rangle, \langle n_2, l_2\rangle, \langle n_3, l_3\rangle \langle n_4, l_4\rangle\}$. The shorthand notations for capturing the nodes and locations in a view would be $N_v = \{n_1, n_2, n_3, n_4\}$ and $L_v = \{l_1, l_2, l_3, l_4\}$, respectively. Finally, the system specification also includes the *number* of faulty nodes as defined in Section 4.6. If our example, if $f = 1$, then at most one out of the four vehicular nodes could behave according the faulty node model.

The model we have introduced in this section is abstract, and fairly general. It allows arbitrarily many nodes and locations, as well as arbitrary sensing patterns between the locations. The model can be extended to include time (i.e., representing a dynamic system) by changing the configuration function to be a mapping $c : L \times T \mapsto N \cup \{e\}$, where T represents the set of time points. We explore this further in the trace based evaluation in Section 7.3.

5 IMPOSSIBILITY RESULTS FOR VIEW VERIFICATION

In this section we will analyse the theoretical bounds (expressed in terms of properties of the sensing graph, number of faulty vehicles, and view size) for when it is at all possible to verify that a view is sound and/or complete. To state the necessary theorems we first introduce the concept of

view compatibility that we use to define verifiability properties of the view in presence of faults. Then we we go on to consider verifiability of completeness and soundness separately, with one theorem for each property.

5.1 View compatibility

A vehicle that receives a membership view can check the correctness of the parts of the view that relates to locations that are covered by the on-board sensors of that vehicle. We define compatibility between a vehicle and a view in the context of a particular system specification and configuration to hold if there is no contradiction between the information in the view and what the vehicle can sense:

Definition 4. *Given a system specification $\mathcal{S}$, and a configuration c, a vehicle node $n_i \in N$ is* **compatible** *with a view v iff $\forall l \in s(n_i) : (c(l) = n \neq e) \leftrightarrow \langle n, l\rangle \in v$.*

In other words, if a location l is covered by the sensors of the vehicle node n_i and there is another vehicle located at l, then there should be a corresponding entry in the membership view for the view to be compatible with n_i. Similarly, if there is an entry in the membership view, and that location is covered by the sensors of n_i, then that entry must be correct.

We can extend the concept of compatibility from one node to all nodes. Verifiability rests on the ability for a vehicle to sense its surroundings together with the knowledge that its neighbours are *also* able to sense their surroundings. However, we have to consider the fact that some (up to f) vehicles might be faulty. Therefore, we define compatibility for an entire configuration as follows:

Definition 5. *Given a system specification $\mathcal{S}$, a configuration c is f-***compatible** *with a view v iff there are at most f vehicles in N_v that are not compatible with v.*

Intuitively, we can think of this definition stating that at most f vehicles can be faulty, meaning that they might not react correctly to inconsistencies between what they should observe and what is stated in the view. However, all the remaining vehicles in N_v should observe facts that are consistent with the view for the configuration to be compatible with the view. If $f > |v|$, then all configurations are f-compatible with v. We will use the concept of f-compatibility to define verifiability with respect to completeness and soundness in the following subsections.

5.2 Impossibility of completeness verification

We now proceed to define the concept of completeness-verifiability which is the idea that it is possible to verify that the view is complete under the assumption that up to f vehicles can be faulty. For this to be true, then there should not exist any configuration of vehicles that violate completeness but which is still f-compatible with the view.

Definition 6. *Given a system specification $\mathcal{S}$, a view v is* **completeness-verifiable** *iff there is no configuration c that is f-compatible with v and for which v is not complete.*

Or put another way: If there is a configuration c so that the view is f-compatible with c, but which makes the view incomplete, then the view is not completeness-verifiable. Note that the verifiability of a view is only defined in relation to the system specification. That is, the verifiability of a view does not depend on the true locations of the nodes (as modelled by the function c).

Now we can state the first main theorem which restricts the cases where a view can be verified to be complete in terms of the connectivity properties of the sensing graph G. In particular, we require that the smallest dominating set in G cannot be larger than the guaranteed number of non-faulty vehicles in the view.

Theorem 1. *Let $\mathcal{S} = \langle N, G, f \rangle$ be a system specification and v a view, where $|N_v| < |N|$. The view v is not completeness-verifiable if*

$$\gamma(G) > max(0, |v| - f),$$

where $\gamma(G)$ indicates the number of vertices in the smallest dominating set.

Proof. We prove this by assuming that the above condition holds and show that the view v is not completeness-verifiable by showing that there exists a configuration that is f-compatible with v, but for which v is not complete. Assume that $\gamma(G) > max(0, |v| - f)$. Let $C \subseteq N_v$ be an arbitrary subset of the vehicles in the view v such that $|C| = max(0, |v| - f)$. Then by the premise we know that there exists a hidden location $l_h \in L$ which cannot be sensed by any vehicle in C ($l_h \notin s(n_i)$ for any $n_i \in C$), since otherwise the locations of the vehicles in C would be a dominating set of size $max(0, |v| - f)$. We now construct a physical configuration c as follows. First, we let $c(l_h) = n_h$, where $n_h \in (N \setminus N_v)$. Moreover, for every vehicle location $\langle l_j, n_j \rangle \in v$ such that $l_j \in s(n_i)$ for some $n_i \in C$, we let $c(l_j) = n_j$, and $c(l) = e$ for all remaining cases (if any). Clearly c is f-compatible with v since all vehicles in C are compatible with c (leaving at most f remaining vehicles in N_v). However, since n_h is not in the view, the view is not complete, and so the view is not completeness-verifiable. □

We illustrate the idea of the proof in Figure 6. The circular nodes represent locations and the lines between them represent the edges of the sensing graph G. The arrows represent a configuration that maps locations to vehicular nodes. The graph has a domination number $\gamma(G) = 3$, since the smallest possible dominating set is of size 3. Now consider the view $v = \{\langle n_1, l_1 \rangle, \langle n_2, l_2 \rangle, \langle n_3, l_3 \rangle \langle n_4, l_4 \rangle\}$. If we assume that up to 2 vehicles might be faulty ($f = 2$), then any subset of $|v| - f = 2$ nodes from N_v will never be able to sense all locations in the graph. For example, if we let $C = \{n_2, n_3\}$, then the location l_h in the graph will be hidden from these two nodes. Therefore a configuration as illustrated with the arrows in the figure would be f-compatible with v, but v is clearly incomplete since $\langle n_h, l_h \rangle$ is not part of the view v.

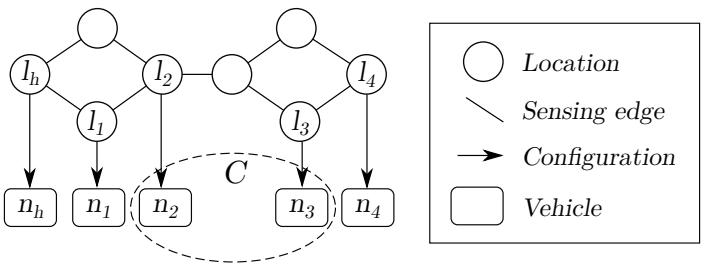


Fig. 6. Illustration of Theorem 1.

Finding the domination number $\gamma(G)$ is NP-hard in the general case. However, many common graph types have closed form expressions for determining the domination number. For example, a 6×6 grid has a domination number $\gamma(G) = 10$, which would mean that even with a single fault, only views with size 12 or larger can be verified for completeness.

Note that the theorem does not provide any guarantees of verifiability if the inequality is not met. The exact conditions when a view can be verified depends heavily on which vehicles that are included in the view. In Section 6 we address this problem through an automated reasoning framework.

5.3 Impossibility of soundness verification

The problem of verifying view soundness is largely analogous to verifying completeness. However, we shall see that the conditions for the impossibility result differs and requires more information with regard to the view locations in relation to the sensing graph G. We define soundness verifiability as follows:

Definition 7. *Given a system specification $\mathcal{S}$, a view v is* **soundness-verifiable** *iff there is no configuration c that is f-compatible with v and for which v is not sound.*

Again, this is the same as saying that if there is some configuration c that is f-compatible with v, but which makes v unsound, then v is not soundness-verifiable.

The second main theorem in the paper states that a view cannot be verified to be sound if the connectivity (as measured by node degree) between locations *within* the view is insufficient.

Theorem 2. *Let $S = \langle N, G, f\rangle$ be a system specification, and v a view. Moreover, let G_v be the subgraph on G induced by the vertices in the view, L_v. The view v is not soundness-verifiable if*

$$\delta(G_v) < f,$$

where $\delta(G_v)$ indicates the minimum degree in the graph G_v.

Proof. We prove this theorem by assuming that $\delta(G_v) < f$ and then show that there is a configuration that is f-compatible with v, and which violates soundness of v (see Definition 7). Assume that the inequality holds and let $l \in L_v$ be a vehicle location with at most $f-1$ neighbours from the set L_v (such a vehicle must exist due to the assumption), and let $F \subseteq N_v$ be the set of vehicles that correspond to l and the (at most) $f-1$ neighbours from the view. Note that $|F| \leq f$ since f can be larger than N_v. Now considering the remaining vehicles n_i in the (possibly empty) set $N_v \setminus F$, we know that $l \notin s(n_i)$. Therefore a configuration for which $c(l) = e$ would violate soundness of v, but still be f-compatible with v since there are at most f vehicles (those in the set F) where l is not compatible with v. □

We illustrate the proof in Figure 7. The location and sensing graph is the same as in Figure 6, as is the proposed view $v = \{\langle n_1, l_1\rangle, \langle n_2, l_2\rangle, \langle n_3, l_3\rangle, \langle n_4, l_4\rangle\}$. Here we can see that the subgraph G_v induced by v becomes a disconnected graph with two components ($\{l_1, l_2\}$ and $\{l_3, l_4\}$). The minimum degree in G_v is 1, meaning that if there are at least two faulty nodes ($f = 2$), it will not be possible to verify soundness. This is shown by finding a node with degree 1, (e.g., n_1) and put it together with all its view neighbours (n_2) in the set F, $|F| = 2$. If the configuration is such that $c(l_1) = e$, then the view is unsound, but it is f-compatible with the view v since there are at most $f = 2$ vehicles that are not compatible with v.

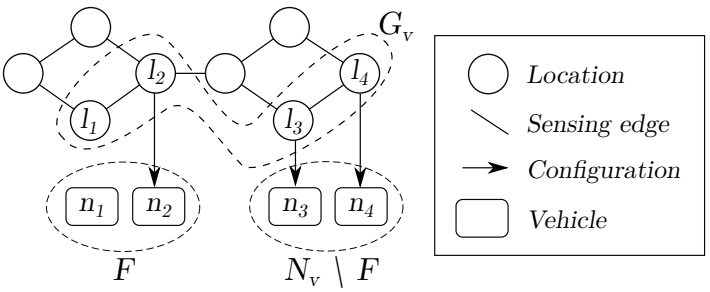


Fig. 7. Illustration of Theorem 2.

This theorem is mainly concerned with the ability of vehicles in a view to sense each other. Therefore, it would make sense for vehicles in a view to coordinate their movements in such a way that degree of G_v is kept above 1. For example, the vehicles in the platoon are only able to sense what is immediately ahead and behind them, the minimum vehicle degree in the sensing graph will be 1, meaning that if the number of faulty vehicles is 2 or larger, it will not be possible to verify view soundness in the general case (this is consistent with what we have observed in earlier work [3] for specific instances). By rearranging the platoon so that the head and tail vehicles are not isolated, this problem would be avoided. Even if this is not possible to do at all times, perhaps this could be done at some regular intervals to ensure that the view is still intact. Further exploration of such regular reconfiguration manoeuvres would be an interesting for future work.

Finally, we note that this proposition can be used to derive another bound which only considers the sensing graph and the bound on the number of faulty vehicles.

Corollary 1. *Let $S = \langle N, G, f \rangle$ be a system specification, and v a view. The view v is not soundness-verifiable if*

$$\Delta(G) < f,$$

where $\Delta(G)$ indicates the maximum degree in the graph G.

Proof. $f > \Delta(G) \geq \Delta(G_v) \geq \delta(G_v)$, where G_v is the subgraph on G induced by the vehicles in N_v. By Theorem 2, the result follows. □

For example, we can tell from this corollary that if the sensing graph G is k-regular, then there is no view that is soundness verifiable if $f \geq k$.

This concludes the theoretical analysis of the verifiability problem. We have derived two impossibility bounds given some particular detector capabilities (as formalised by the sensing function $s(n)$). In Section 7.3.2 we discuss and analyse the impact of these constraints in urban junction-based scenarios. In the next section we consider what can be done when verification is not impossible according to these bounds.

6 CONSTRAINT-BASED VERIFICATION OF GROUP VIEWS

In this section we describe how this formalism can be translated to an on-line model, for the cases where view verification can be done. The purpose of this model is to be able to integrate it into a runtime environment and feed it with the information that is known by the vehicle where it is running. The purpose being that in the cases where verification is possible, the framework should be able to inform the other parts of the coordination logic that this view can indeed be trusted. The framework is based on the idea presented in Figure 4 in Section 4.1, where a verifier takes as input a model of the environment, the membership protocol, and the detector and outputs a decision on whether the view can be verified or not.

In the remainder of this section we describe the constraint solver mechanism that we use in the formal reasoning. Following this, we provide pseudocode for the detection and verification process and finally, in the last part of the section, we discuss the implications of having a discrete model of vehicle locations.

For this part of the work we make the additional assumption that $f < |v|$. This will guarantee that at least one vehicle in the view is non-faulty. This is the vehicle at which we consider that the online verification process is run.

6.1 Constraint satisfaction framework

We use a solver for Satisfiability Modulo Theories (SMT) to perform the deductive reasoning in the verification step. We use a subset of the modelling language that includes predicate formulas, integer

numbers, equality, inequality, and universal quantification (but not existential quantification). In particular, based on the model introduced in Section 4 we create a constraint model M which can be represented as a 4-tuple:

$$M = (N, L, \mathcal{F}, C),$$

where N and L are the vehicle and location sets. The set $\mathcal{F}$ contains a number of uninterpreted functions (which is the same as predicates). This includes the configuration function c, the sensing function s, as well the empty location constant e, and the contents of the view v (a constant can be seen as a predicate with arity 0). Finally, the set C contain the constraints that provide semantics to the model by restricting the possible interpretations. We implement the SMT formulation of the model using Z3Py which provides a Python API to the Z3 v4.5.1 theorem prover [11].

The challenge to finding the appropriate constraints for view verification is to manage the epistemological problems relating to what is known, what is believed, what would other entities know/believe, and what are possible variations of reality that meets the current set of known facts. We are not aware that this has been previously studied in the context of dynamic membership views. The fact that the verification should be done on-line means that it should be performed from the perspective of one of the vehicles (called the ego vehicle). Of course all vehicles will perform the same reasoning, so this does not imply a centralised approach. The knowledge model at the ego vehicle should contain the following layers.

- Physical reality. There are some facts known about the physical reality such as the set of locations, as well as those aspects of reality which can be directly observed. Moreover, there are also aspects of physical reality which are not known, such as the location of vehicles outside the sensing range. These must me modelled as open variables.
- Information from and about other vehicles. Claims made by other vehicles are facts in the sense that the claims have been made. Whether or not they should be trusted depends on what the ego node should believe about the trustworthiness of other vehicles.
- Ego vehicle constraints. Basically, each vehicle knows that itself is not faulty, that it should be part of the view, and which its own location is.
- What other vehicles might know or believe. As opposed to the information provided by other vehicles, these variables must contain the supposed knowledge by other vehicles about their surroundings and also how they would have communicated if they had seen something and were not faulty. This also must include information of under which conditions another vehicle would accept a given view.

While a full description of all constraints in the model would require a much longer paper, we will go through these four layers and provide some insight into the main challenges in each. While translating predicate logic to SMT constraints is not in itself hard, it is sometimes difficult to find a formulation that makes the problem solvable by the theorem prover. Since our model includes infinite domains, the general problem is actually undecidable. The full model contains approximately 1600 constraints. The exact number depends on the the particular scenario (such as the number of edges in the sensing graph).

6.1.1 Physical reality. In order to model physical reality we use the configuration function $c : L \rightarrow N \cup \{e\}$ introduced in Section 4.2. However, in the SMT model, simply using this function leads to the theorem prover not terminating in the search for a model that satisfies the constraints. A simple solution to this problem turns out to be to introduce the inverse of c which maps vehicles to locations and forcing these two functions to be consistent in the relevant cases (but not in all locations).

When creating these models it is also important to rule out cases which a human can easily identify as spurious, but which a theorem prover will often find unless specifically removed. For example, that no two vehicles can be in the same location, or that a vehicle can only be in a single location at any point in time. To illustrate how these constraints are formalised, we describe the latter of these in full:

$$\forall l_1, l_2 \in L : (c(l_1) \neq c(l_2)) \vee (c(l_1) = e) \vee (l_1 = l_2)$$

This formula states that for all locations l_1 and l_2 in the set of locations L, either of the following three conditions must hold:

- The vehicle at l_1 is not the same as the vehicle at l_2 (as encoded by the configuration c)
- The location l_1 is empty
- The locations l_1 and l_2 are identical

Translating this formula to Z3Py code is straightforward as seen below.

```
def singleLocation():
    l1 = Const('l1', Location);
    l2 = Const('l2', Location);
    return [ForAll([l1,l2], Or(c(l1) != c(l2), c(l1) == e, l1 == l2))];
```

There are several other sets of constraints that relate to the physical environment. This includes expressing the sensing graph as sensing constraints and ensuring that empty locations are modelled appropriately. Moreover, since vehicles can be located outside the area of interest, we must manage two different location sets. An infinite location set where nodes can exist outside the area, and the finite location set L that contain all the locations in the area of interest.

6.1.2 Information from and about other vehicles. An important part of the model in this regard is the view $v = \{\langle n_1, l_1\rangle \ldots \langle n_{|v|}, l_{|v|}\rangle\}$. To make the solver handle this construction, we add a number of functions and constraints that tests view membership for both locations and vehicles, define the soundness and completeness predicates according to Section 4.5, as well as some basic view consistency constraints for views. While we have presented views a set of pairs in this paper, the modelling of sets is far from straightforward in a constraint solver since this often requires the use of the existential quantifier. Instead, it is much more efficient to model these as finite lists, but this requires additional constraints to rule out duplicate elements, to manage the order of elements, and avoid empty view elements.

The ego vehicle also needs to keep track of which other vehicles it believes to be faulty or non-faulty. This information is important when modelling what the other vehicles should know, since even if a faulty vehicle would know about some inconsistency it would not communicate this fact. One must also bound the number of *believed* faulty vehicles (corresponding to the variable f). Again, for efficiency of the solving procedure, the best way to manage this is to put all vehicles believed to be faulty in a list structure.

6.1.3 Ego vehicle constraints. Recall that the purpose of the ego vehicle is to represent the vehicle at which the verification procedure is executing. This means that more information is known about this particular vehicle than about the other vehicles. There are three special constraints that we add with regards to the ego vehicle as given below.

- The ego vehicle is in some location ($\exists l \in L : c(l) = \text{ego}$)
- The ego vehicle is in the view ($ego \in N_v$)
- The ego vehicle is not faulty ($\text{ego} \notin F$)

These constraints pose no particular difficulty for the solver. Worth mentioning perhaps is that the first of these constraints is actually translated to a large disjunction of the following form: $(c(l_1) = ego) \vee (c(l_2) = ego) \vee \ldots \vee (c(l_{|L|}) = ego)$. Since existential quantifiers and set inclusion should be avoided in the SMT model if possible.

6.1.4 What other vehicles might know or believe. Finally, the most interesting aspect of the verification model is that the ego vehicle should keep track of what the other vehicles would know about their surroundings, and whether they should accept the view or not.

We introduce the predicate $a(n, v)$ which is true if vehicle n *accepts* the view v. A non-faulty vehicle will only accept a view if it is a proper view (according to Definition 1) and the vehicle is compatible with the view (see Definition 4). Note that the constraints only apply to vehicles in the set of non-faulty vehicles. So a faulty vehicle can very well accept an incomplete or unsound view.

We have already discussed the view constraints in Section 6.1.2. However, the translation of Definition 4 to SMT deserves some more discussion. This definition states that a vehicle node $n_i \in N$ is compatible with a view v iff $\forall l \in s(n_i) : (c(l) = n \neq e) \leftrightarrow \langle n, l \rangle \in v$. To make this requirement tractable for the solver, we create the following three constraints.

- A non-faulty vehicle only accepts a view where the locations in the view are known by that vehicle not to be empty.
- A non-faulty vehicle only accepts a view if identities and locations match what is known by that vehicle.
- A non-faulty vehicle only accepts a view if includes all non-empty locations known by that vehicle.

The reason for splitting the constraint in three is that we need to care for the inverse of the configuration function (that maps nodes to locations), as well as the functions that models known empty locations.

When added to the full model, all these constraints together with the variables controlling potential locations of vehicles, and whether they are faulty or not will enable an the ego vehicle to enumerate all possible configurations that match the known facts. In a sense, this can be seen as a search problem where the task is to place faulty vehicles in the location set so that they are able to violate soundness or completeness of the view without being discovered by a non-faulty vehicle. If the search fails, then we know that there can be no faulty vehicles in the area. In the following section, we describe the algorithms used to perform this search.

6.2 Detection and verification mechanisms

In Section 4.1, we presented an architecture for secure membership views. In this section we describe how the detection and verification mechanisms are integrated and implemented using the formal framework we have introduced above. In listing 1 the same logic as was presented in Figure 4 in Section 4.1 is presented as psuedocode. The input to this view analysis procedure is a view v, a property *prop* that is either soundness or completeness, and a vehicle n_i at which the analysis is performed.

Basically, this algorithm first checks if a violation of the property can be detected, or if the property can be verified, or otherwise returns that there is not enough data to say for sure whether the view meets the property or not. We describe each of the two subroutines for detection and verification below.

The high-level pseudocode for detecting a violation is shown in Listing 2. The detection algorithm is a distributed algorithm that runs in all the vehicles that are part of the view. First, each vehicle determines based on locally available information whether it detects any violation of a property. For incompleteness, this means that it senses a vehicle that is not part of the view, and for soundness,

Listing 1 Pseudocode for view analysis

```
1: procedure AnalyseView(v, prop, n_i)
2:     if DetectViolation(prop, v, n_i) then
3:         return ViolationDetected
4:     else if Verifiable(v, prop) then
5:         return PropertyVerified
6:     else
7:         return InsufficientData
8:     end if
9: end procedure
```

it means that one of the vehicle-location-pairs in the view does not match with the sensors of the vehicle.

Listing 2 Pseudocode for to detect violation of property *prop* of view v at vehicle n_i

```
1: procedure DetectViolation(v, prop, n_i)
2:     if v violates property prop then
3:         BroadcastViolationDetection(prop)
4:         return True
5:     else
6:         return ReceiveViolationDetection(prop)
7:     end if
8: end procedure
```

If a violation is detected, this information is broadcast to all other vehicles in the vicinity. The final step of the algorithm is to receive any such violation detections from other vehicles. Note that according to our fault model in Section 4.6, a violation detection from a non-faulty vehicle will eventually reach all other non-faulty vehicles. However, a faulty vehicle could also make the other vehicles detect a spurious violation. Sorting out correct alerts from false alerts reduces to an instance of Byzantine fault tolerance, which is outside the scope of this work.

Listing 3 shows the pseudocode for the verification procedure given a view v. Essentially, the system model M is initialised with the constraints that form the system model and fault model (line 2). Moreover, the view acceptance conditions described above are added to the model (line 3).

Listing 3 Pseudocode for the verification procedure given a view v

```
1: procedure Verifiable(v, prop)
2:     M ← {System model, including fault model}          ▷ Section 4
3:     M ← M ∪ {View acceptance conditions}               ▷ Section 6.1
4:     if M, v, prop ⊨ False then
5:         Raise exception
6:     else
7:         return (M, v, ¬prop ⊨ False)
8:     end if
9: end procedure
```

The model M together with the assumptions that the view is sound and complete should now have at least one solution that fits with all the constraints. The constraint solver is invoked to check this fact. If no solution is found (line 4), then an exception is raised, since violation of the property should have been caught by the detection mechanism.

The next step (line 7) is to first assume that the view property is not met and try to find an interpretation that fits with this assumption. If such a solution is found (i.e., the model *does not* entail False), then we know that there is a possible scenario where the view has been accepted by the non-faulty vehicles despite violating the property. The reason this can happen is because some vehicles can be faulty and provide false information to the non-faulty vehicles. On the other hand, if no solution can be found to the constraint satisfaction problem (i.e., the model entails False), then we know that the view must meet the desired property (given the assumptions of the model).

6.3 Location modelling

In the model we use, the physical area is composed of a finite number of locations, and each vehicle can only be in one location at a time. This abstract model is adequate in relation to verifying view soundness since the information needed is a finite list of locations of the vehicles in the view. In this case, a discrete sensing graph can be generated dynamically based on known properties of the sensor characteristics, the environment, and the list of locations in the view.

When it comes to view completeness, the situation is more complex. In this case there is no obvious algorithm that would generate a correct and unique set of locations and corresponding sensing graph. The physical locations cannot be easily discretisised, and the vehicle sizes can also vary considerably. However, we can take advantage of the fact that the entire area *should* be empty except for the vehicles that are present in the view. Therefore, in addition to the locations generated by the view set, we can generate supposedly empty locations by assuming a maximally packed area using vehicles of minimal size.

A large vehicle (e.g., a truck) outside the view would then potentially occupy a large number of locations (thereby violating the assumption of one location per vehicle). However, from the modelling perspective, since the truck violates view completeness (since it is not part of the view), this is simply considered as multiple vehicles violating completeness.

7 EVALUATION

In this section we evaluate the performance of the constraint-based verification approach. We organise this section by first presenting the main metrics that we use to measure the detection and verification performance. The remainder of the evaluation is composed of two parts, (i) an evaluation based on synthetically generated scenarios that allows studying fundamental parameters relating to the model structure, and (ii) a trace-based evaluation where vehicular mobility traces are used to assess how the approach would perform in a more realistic setting (including studying effects of stale views and node mobility).

7.1 Metrics

A core idea in this paper is to make a distinction between the cases where a violation cannot be detected due to insufficient data and the cases where a violation cannot be detected because we know that it does not exist. Recall the possible outputs from the view analysis procedure in Section 6.2, with respect to some view property (i.e., soundness or completeness).

- ViolationDetected, a violation is detected
- PropertyVerified, a violation can be ruled out
- InsufficientData, neither detection or verification can be done

The three outcomes of the detection/verification together with the two possible ground truths (no violation / violation) result in six possible cases for each correctness property as shown in Table 2. Since we have two view correctness properties (soundness and completeness), there are a total of $6 \cdot 6 = 36$ possible outcomes for each case (e.g., we might accurately detect a soundness violation, but fail to verify completeness). However, we will consider the results relative to the two properties separately in order to reduce the complexity of the presentation.

Table 2. Possible outcomes of the combined detection and verification

	No violation	**Violation**
ViolationDetected	False Detection (FD)	True Detection (TD)
PropertyVerified	True Verification (TV)	False Verification (FV)
InsufficientData	Missing Verification (MV)	Missing Detection (MD)

Ideally, we want all cases to end up as either true verifications (TV) or true detections (TD) depending on whether a violation has occurred or not. If the framework fails to detect violation or verify a view due to insufficient information, this will manifest as a missing detection (MD) or missing verification (MV). However, since the model we deploy is a deterministic one, we would normally expect no cases of false verification (FV) or false detections (FD). If that occurs, then at least one of the assumptions in the model does not hold in the given scenario (we will see that happen if for example, the assumed number of faulty vehicles is incorrect). Obviously there is a connection to the metrics used in binary classification in machine learning applications (i.e., true/false positive/negative).

7.2 Synthetic scenario evaluation

We now present the first part of the evaluation which is based on purely synthetic scenarios. First we describe how the test scenarios are generated, then present the results based on four different aspects: sensing probability, fault probability, robustness, and scalability.

7.2.1 Scenario generation. To evaluate the effectiveness of the verification approach we evaluate 400 different synthetically generated scenarios. Unless otherwise stated, we base the scenarios on an area with 20 possible vehicle locations. Depending on the inter-location distance, this corresponds to a physical area of approximately 200-300 m^2. The sensing graph in each synthetic scenario is based on a *connected* Erdős-Rényi graph with 20 locations and edge probability p. We exclude disconnected graphs since both detection and verification are basically meaningless in these cases. We instantiate a group membership view with 5 vehicles and their corresponding locations. We differentiate between three types of scenarios:

- Benign scenarios, where all vehicles are non-faulty and the view accurately reflects the scenario.
- Unsound scenarios, where f_r of the vehicles in the view are not located where they should be according to the view v.
- Incomplete scenarios, where there is one vehicle n_i that is not part of the view v, and $f_r - 1$ vehicles in the view that will pretend not to sense n_i.

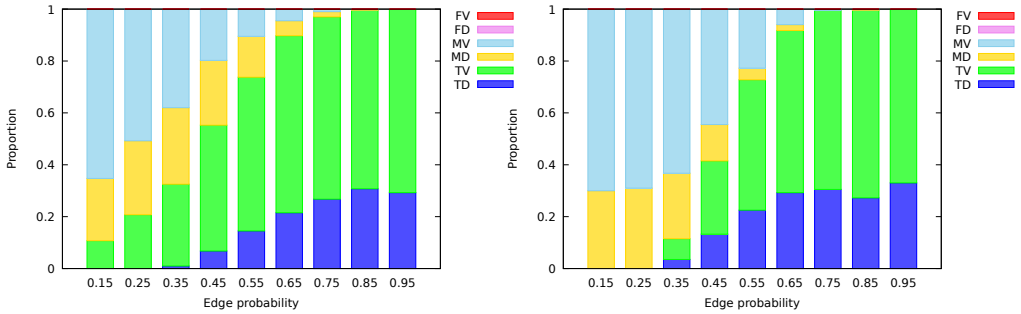
The locations of the faulty nodes in are chosen so to maximise the distance to the non-faulty nodes in the view. In each run, the probability of creating a faulty scenario is $P_f = 0.3$. We will restrict the number of faulty nodes (real and assumed) to 1 on most of the experiments. A summary of the default parameters for the scenarios are shown in Table 3.

Table 3. Default parameters

Description	Symbol	Value
Number of locations	$\|L\|$	20
Number of vehicles	$\|N\|$	unbounded
View size	$\|v\|$	5
Edge probability	p	0.55
Real number of faulty vehicles	f_r	1
Assumed number of faulty vehicles	f_a	1
Fault probability	P_F	0.3
Number of scenarios in each run		400

7.2.2 Sensing probability. The first experiment we perform concerns how the ability to verify correctness of a view depends on the ability to sense other vehicles in the surrounding locations. This is represented by the edge probability parameter p in the sensing graph generation. It is natural to expect that the verification rate increases with improved sensing capabilities of the on-board sensors. However, what is not obvious before performing these experiments is how the ability to verify view correctness relates to the ability to detect faults.

Figure 8a and 8b show the 6 different outcome categories for an experiment where the edge probability p ranges from 0.15 (very sparse, but still connected, graph) to 0.95 (very well-connected graph). In Figure 8a the faulty views are unsound, whereas in Figure 8b the faulty views are incomplete.



(a) Including views that violate soundness (b) Including views that violate completeness

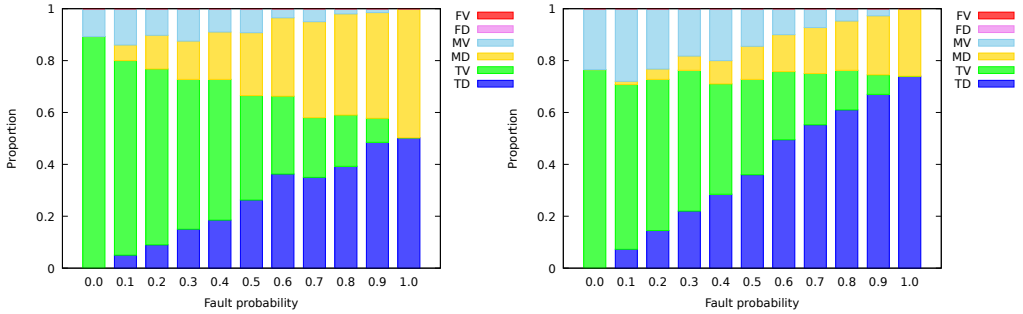
Fig. 8. Proportion of outcomes as a function of connectivity of the edge probability in the sensing graph G

The first thing to note about these results is that there are no false detections or false verifications. This is due to the way the model has been set up. The detector only raises an alert when its sensors detect a deviation from what is stated in the view. Moreover, the detector is assumed to run in a non-faulty node so if the view does not correspond to the environment according to its sensors, there is clearly a violation. However, as we shall see later on, if the assumptions of the verifier is incorrect, false verifications can and will occur.

The other thing to note is that the verifier performs well compared to the detector for view soundness. Recall that the verifier and detector has the same sensing capabilities (i.e., ability to know which vehicles are in the vicinity). Consider for example Figure 8a when the graph connectivity factor is 0.45. The detector detects only a small fraction of the violations (dark blue vs yellow), but

the verifier is able to correctly verify around three quarters of the correct views (green vs light blue). For completeness (Figure 8b), the opposite is true. It is easier to detect a violation, than to verify correctness.

7.2.3 Fault probability. The next experiment looks at how the fault probability parameter P_f affects the detection and verification of views. Figures 9a and 9b show the performance of the view analysis as the fault probability ranges from 0 to 1. As expected, the number of detections increases and the number of verification decreases as the fault probability increases. However it is also interesting to look at the proportion of the cases that are correctly categorised (TV+TD) versus the cases where there is insufficient data (MV+MD). In Figure 9a this ratio decreases as the fault probability increases. The reason for this deterioration is that the verifier is comparatively better at verifying correct views than the detector is to detect violations (as we discussed in the previous subsection). Therefore, as the number of faulty cases increase, the performance gets worse.



(a) Including views that violate soundness (b) Including views that violate completeness

Fig. 9. Proportion of outcomes as a function of fault probability

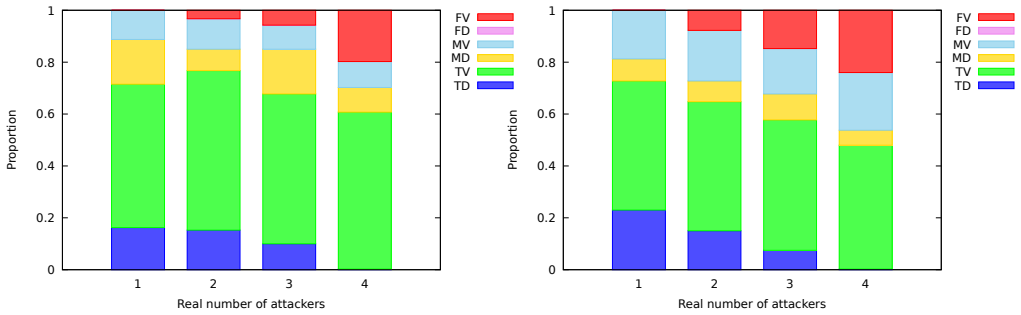
On the other hand, in Figure 9b, the ratio between correct classifications and the cases where there was insufficient remains the same as the fault probability changes.

7.2.4 Robustness. The purpose of this experiment is to study how making too optimistic assumptions in the system model affects the ability of the reasoning framework to perform adequately. Figures 10a and 10b show the result for soundness and completeness when increasing the *real* number of faulty vehicles (f_r), but keeping the assumption in the verification procedure that there is at most one faulty vehicle node.

We can see in both figures that the system falsely verifies views that violate soundness or completeness. This is an expected result. As the real number of faulty vehicles increase to 4, most of the incorrect views are falsely verified correct. Moreover, the the number of true detection decrease correspondingly. As a larger portion of the view is controlled by faulty vehicles (that do not accurately report what they see), the ability to detect a violation decreases.

7.2.5 Scalability. The results presented so far are all based on scenarios with a constant size of the location set $|L| = 20$. This parameter significantly impacts the number of constraints in the constraint satisfaction problem associated with view verification. Therefore, it is interesting to investigate the scalability of the verification mechanism using larger location set sizes.

We perform this experiment by varying the size of L from 20 to 140 locations. We keep the other parameters constant, including the edge probability which means that the number of edges increases quadratically with the number of locations. For each size of L we generate 100 different



(a) Including views that violate Soundness (b) Including views that violate completeness

Fig. 10. Proportion of outcomes as a function of the real number of faulty vehicles

scenarios and measure the time taken to either detect a violation or verify the view correctness. The experiments are run in a high-performance cluster at Linköping University, but each experiment is run as a single core job on a Intel Xeon Gold 6130 processor.

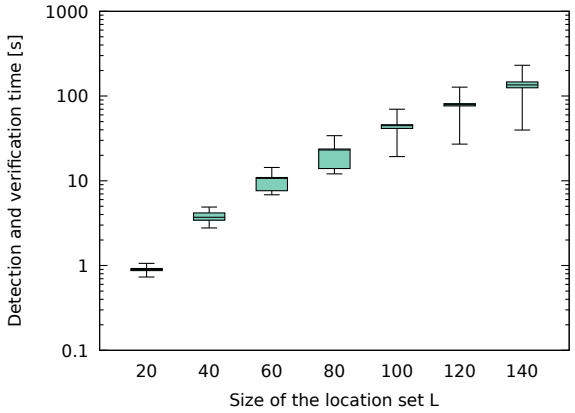


Fig. 11. Time taken for detection or verification for different sizes of L

Fig 11 shows the results of the execution time experiment using boxplots and a logarithmic y-axis. Each box represents all data points between the lower and higher quartile. The whiskers represent 95% of all data points. Both the mean value and variation increase rapidly as the size of the location set increases, but appears to be subexponential. In terms of applicability for real-world scenarios, it is clear that some kind of performance improvement would be required. The smallest location sizes are no problem, but having around 100 locations takes almost a minute on average. Note that the code has not been optimised for speed in any way, so there is probably room for improvement in this regard.

7.3 Trace-based evaluation

The purpose of this second part of the evaluation is to investigate the applicability of the proposed framework using more realistic scenarios. In particular, we study how static road data together with dynamic vehicular mobility traces can be used to generate the models used for view verification

and investigate the properties of these models. Moreover, we evaluate the detection and verification performance in these trace-based models.

7.3.1 Scenario generation. To generate the scenarios, we use data from the vehicular mobility traces for the city of Cologne in Germany [37]. These traces have been created using the Sumo microscopic mobility simulator and have been thoroughly validated against real measurements. This data set is one of the best available vehicular mobility traces since it provides second level updates and provides locations for all vehicles in a large area (as opposed to a small subset, as given by for example traces from taxis or buses).

We selected a subset of the full 24h Cologne traces made available from the Cologne project website [1]. We extracted the traces from one hour starting at 07:00 in the morning that were within 2km from the city centre. This resulted in a 570MB text file. In order to make the data useful in our proposed model, it was necessary to find even smaller units for which it makes sense to form groups for reliable communication.

For an urban environment, the most obvious focus point for coordination actions is an intersection. Thus, we used the road data information in form of the Sumo configuration files to extract all the intersections in the area of interest. In sumo, these are named *junctions*, and we will use this terminology in the rest of this section. However, this concept not only covers regular intersections, but all cases where two or more lanes meet. This extraction resulted in 513 unique junctions.

For each junction, we then collected the vehicular mobility traces occurring within 40m from the centre of the junction. We discarded those junctions where there was no traffic at all during the selected hour, resulting in 420 junctions that were further processed to create the location sets.

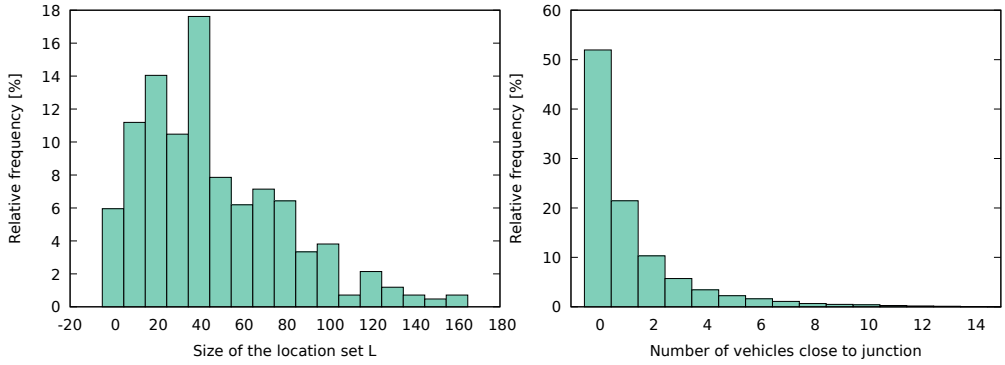
The vehicular traces around each junction, which basically form a set of (x,y)-locations, could in theory be used to directly create the location set L. However, this results in thousands of unique locations for each junction, resulting in a computationally intractable problem. Therefore, we kept only those locations whose distance was more than 2m (corresponding to the approximate width of a car) from any previously recorded location.

7.3.2 Properties of junction models. We now investigate the resulting models of the junctions to better understand the data used in the remainder of the evaluation. First, in Fig. 12a the histogram of the location set size for each junction is shown. Most of the junctions have 40 or fewer possible locations, but some very complex ones have as much as 160. Apart from issues with the verification time this could cause (see Fig. 11), perhaps such complicated intersections would also require other coordination mechanisms.

Another important property of the data traces that significantly impacts the approach in this paper is the size of the view set, or put another way, the number of vehicles that are in the vicinity of a junction at a given point in time. We evaluate this by considering every second from 07:00:00 to 07:59:59 and counting the number of vehicles close to each junction in the data set at that time. The resulting histogram over all junctions and time points is shown in Fig. 12b. Note that while there is a variation over time where the vehicles are located, the shape of this distribution remains roughly the same over time. We can see that for most junctions and time points, there are no vehicles at all, and that it is relatively rare for any junction to contain more than 10 vehicles within 40m of the junction centre.

Based on this result and the impossibility bounds on verifiability given by Theorems 1 and 2 in Section 5, it is also reasonable to ask to what extent verifiability will be even theoretically possible in in the generated junction models. The short answer is that this will depend on the sensing capabilities of the vehicles (as modelled by the sensing graph G). Therefore, we will now

[1] http://kolntrace.project.citi-lab.fr/

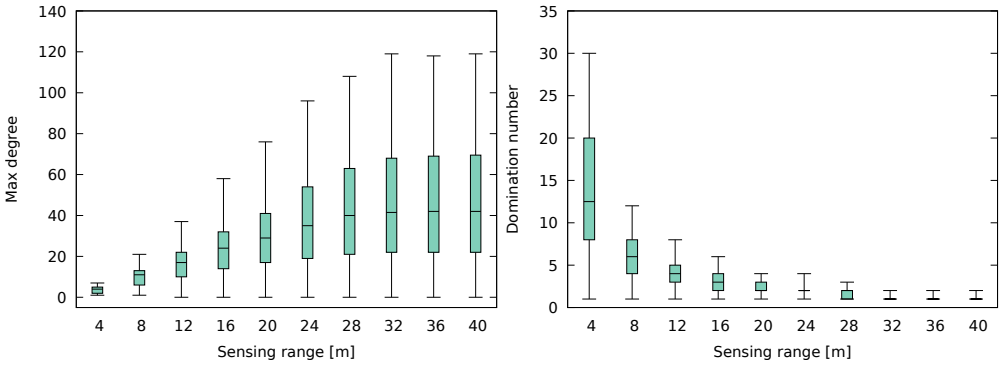


(a) Location set size of the sampled junctions (b) Number of vehicles at the junctions

Fig. 12. Junction histograms

consider how varying the sensing capabilities (modelled as a simple sensing distance) impacts the parameters used in the theoretical framework.

Regarding view soundness (i.e., that all the vehicles in the view are located where they claim to be), Theorem 2 states that if the minimum degree in the graph induced by the view is less than the hypothesised number of faulty vehicles, it will be impossible to verify soundness. However, without knowing the contents view, this theorem is not directly useful to assess the properties of the junctions. The corollary to the theorem is view-independent and considers the *maximum* degree in the entire sensing graph. Fig. 13a plots how this number is affected when varying the sensing range of the vehicles. The average results indicate that for most of the time, this bound will not cause any problems with regards to the verification (having more than 10 faulty nodes in single junction must be considered unlikely). However, the lower 95% whiskers also point to the fact that for a non-negligible proportion of the junctions, soundness will not be verifiable, even with a very long sensing range and a single faulty vehicle.



(a) Max degree vs. sensing range (b) Domination number vs. sensing range

Fig. 13. Properties of the sensing graph of junctions with varying sensing range

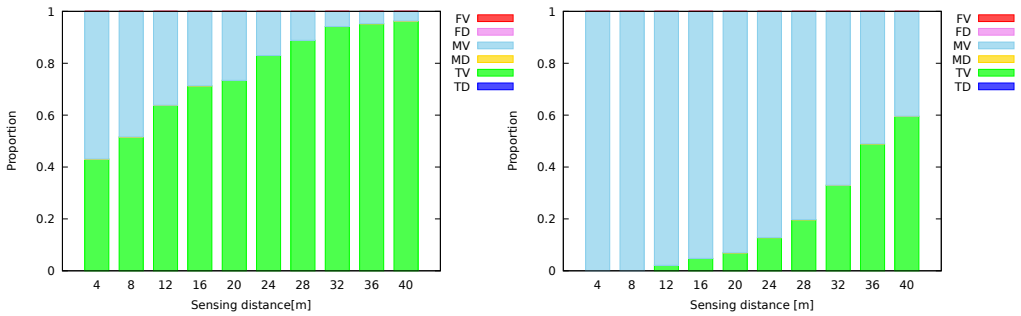
For completeness (i.e., that the view really includes all vehicles in the neighbourhood), Theorem 1 states that if the domination number of the sensing graph is larger than $|v| - f$ where $|v|$ is the

view size and f is the hypothesised number of faulty vehicles, then it will be impossible to verify completeness of the view. Fig. 13b shows how the domination number of the junctions in this study is affected by varying the sensing range. Given the results from Fig. 12b, we know that fewer than five vehicles is often the norm in a junction, so if the view size must be larger than 5 vehicles, there is little point in trying. Passing this threshold seems to occur reliably somewhere around a sensing distance of 20m (corresponding to the length of four vehicles). Of course, this is just a lower bound, so it is still necessary to evaluate the actual performance. Moreover, in most cases the number of vehicles in the intersection is even less than this (e.g., just one or two).

To summarise, these experiments revealed that for most of the time, the impossibility theorem on soundness verification will not be an issue for most cases, whereas the theorem that limits the ability to verify completeness is likely to cause problems if the view size is smaller than 5.

7.3.3 View verification performance. The purpose of the next experiment is to apply the proposed constraint-based framework on the trace-based junction models. To perform this step, we must first map the location of the vehicles entering the junction to the locations represented by L. Since the locations in L are typically just 2m apart, this is simply a matter of finding the closest location.

Figure 14a shows the verification performance of the proposed approach aggregated over all the junctions in the area at a given time point (07:00). Note that all views in this experiment are both sound and complete (we use the actual locations of vehicles to generate them). Therefore, there are no detections, only true verifications (TV) in light green and missing verifications (MV) in light blue. This confirms the indication from the previous experiment that soundness is indeed possible to verify in a large proportion of the cases. With a sensing distance of 20m, more than 70% of all views can be verified.



(a) Proportion of views verified for soundness (b) Proportion of views verified for completeness

Fig. 14. View verification performance

Unfortunately, verifying completeness turns out to be more difficult as can be seen in Fig. 14b. A sensing distance of at least 36m is required to verify more than half of the views. This is not an unreasonable distance given free line of sight, but in an urban environment this is probably not feasible. It is possible that this problem can be ameliorated by considering other ways to reduce the sensing graph (and thereby improving the chance that the nodes in a view are able to sense the entire graph). For example, we have made no distinction between lanes that enter a junction, and lanes that exit the junction. Vehicles that coordinate access to an intersection would probably not need to consider the latter category. Exploring such optimisations would be an interesting extension of this work.

7.3.4 Impact of view staleness. Finally, the last experiment presented in this section concerns the dynamic behaviour of vehicles in an intersection and the fact that views change over time. Moreover, we use these dynamic changes to model a large fraction of the vehicles as faulty. Faulty vehicles will not accurately report their location, nor will they communicate with the non-faulty vehicles.

We generate the scenarios by considering a specific junction (named 2111201731) during 600 time point pairs, t and $t + d$ where d is the delay between these. The view to be analysed is created based on the locations of vehicles at time t, whereas the real soundness and completeness of the view is evaluated at time $t + d$. Therefore, any changes of the vehicle locations between t and $t + d$ will cause the view to be unsound, or incomplete, or both.

When considering each time point pair, there are four categories of vehicles.

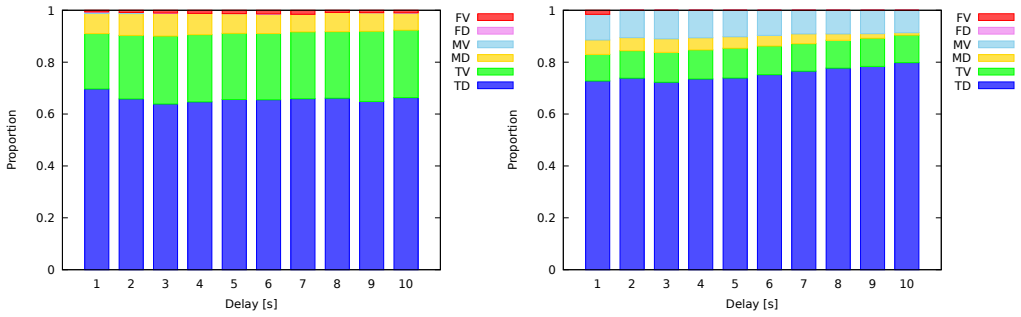
- The vehicles that are located near the junction at time t but not at time $t + d$ (i.e., they have left the junction). These are modelled as faulty vehicles that are part of the view, but which do not communicate with the non-faulty vehicles.
- The vehicles that are located near the junction at time $t + d$ but not at time t (i.e., they have just entered the junction). These are modelled as faulty vehicles that are in the area but not part of the view (if they were malicious, they would be hiding, but this behaviour can also be caused by packet loss).
- Vehicles that are close to the junction at both time t and $t + d$, but which have changed their location. These are also modelled as faulty vehicles, not having reported their location change. Again, such behaviour can be either caused by a malicious entity, or by problems with sensors or communication.
- Finally, vehicles that have the same location at time t and time $t+d$ are modelled as non-faulty vehicles. These can communicate with each other and also sense their surroundings.

Modelling all vehicles that have changed between the two time points as faulty is of course quite pessimistic, it will often result in a large fraction of the nodes being faulty. We can consider this as a sort of stress test of the system. Most views will be unsound and/or incomplete, and the interesting question is whether this will be detected or if perhaps some of them will be falsely verified. To generate the sensing graphs we assume a sensing distance of 20m. Moreover, we discard all pairs of time points where there is no single non-faulty node since otherwise there is no vehicle that could perform the view verification procedure.

In Fig. 15a the results relating to view soundness are shown. The delay on the x-axis correspond to the value of d. As expected, a large fraction of the views are unsound, and we note that most of these are also detected by the non-faulty nodes. For the views that are sound almost all of them are verified. However, we also note that there are some false verifications (coloured red). The reason is similar to what caused the false verifications in Section 7.2.4, that the number of real faulty vehicles exceed the assumed limit $f = 1$.

Finally, in Fig. 15b, the outcome of detecting and verifying view completeness is shown. Again, most views are not correct, and most of these cases are detected. However, of the complete views, only around half of them are verified to be complete. On the positive side, there are fewer views being incorrectly verified as being complete.

With this we conclude the evaluation section. We have evaluated the proposed approach both using purely synthetic data to be able to fully control all parameters such as graph size and connectivity, as well as using vehicular mobility traces to assess the applicability in realistic scenarios. As expected, we can see that the ability to verify views is heavily dependent on the connectivity of the sensing graph. But this is equally true for the ability to detect problems in the



(a) Detection and verification of soundness

(b) Detection and verification of completeness

Fig. 15. Detection and verification in presence of delays that cause views to be stale

views, and even if it will never be possible to verify group membership views in every case, these results indicate that at least there are many situations where this could indeed be possible.

8 CONCLUSIONS

In this work we study the feasibility of verifiable group membership views. We have used an abstract formal model of a group view scenario where the participating vehicles must decide whether a view that specifies the location of each vehicle is correct.

We define two basic properties of a view that should be met at all times: soundness and completeness. Soundness states that all vehicles in a view should be located where the view claims they are located. Completeness states that all vehicles in a particular area must be included in the view. Ideally, a vehicle that receives a view should be able to verify that it is both sound and complete. However, as we have shown in Section 5, the ability to verify these properties is severely restricted unless the graph that represents the sensing abilities of the vehicles have very good connectivity properties. In particular, the smallest dominating set must be smaller than the number of non-faulty vehicles in the view for the view to be verifiable for completeness. To verify soundness the vehicle degree within the view must be larger than the number of faulty vehicles.

The cases where the connectivity of the sensing graph is sufficiently high, the ability to verify correctness depends on the particular scenario (i.e., where the view members are located in the graph), and must be analysed on a case-by-case basis. We present a formal framework that uses an SMT-solver to analyse each such case from the local perspective of one node. We have demonstrated that in many cases the ability to verify correct views is on par with the ability to detect violations (using the same sensing and communication capabilities). Moreover, using trace-based scenarios from the Cologne project we could demonstrate that in many cases, view verification is also possible in more realistic scenarios. We also identified some challenges that need further study. This includes the execution time for verification when the location set is large, and the ability to verify view completeness without requiring very long range sensing capabilities.

Relating back to Section 3, where three ITS scenarios were discussed from the perspective of faults and attacks that could cause problems, we now briefly return to these to discuss to what extent the approach proposed in this work would ameliorate these problems (considering what we have learned from the evaluation). Of course without more details we cannot say for sure that they could all be avoided, but it is clear that if a view detection and verification mechanism was in place, the involved actors would at least have a higher chance of making an informed decision. Unless

parameters are badly chosen, the incorrect views in these scenarios would not have been verified since these possible faults would have been found by the constraint solver.

We believe that this work can serve as a first step towards a new paradigm in the design of trustworthy coordination algorithms for future transportation systems. Our main hypothesis has been that the foundation on which group decisions are made (i.e., who belongs to the group, and where are they located), in safety-critical cyber-physical systems must be *verified* to be correct before any decisions are made that involve the safety of the users. We have demonstrated the feasibility of this approach in an abstract setting, but also in close-to real world scenarios.

9 ACKNOWLEDGEMENTS

The author would like to thank the anonymous reviewers for their suggestions which have helped improve the quality of the paper. The author has been partially supported by RICS: the research centre on Resilient Information and Control Systems[2] financed by Swedish Civil Contingencies Agency (MSB), and Centrum för industriell informationsteknologi (CENIIT), project 14.04.

REFERENCES

[1] Y. Amir, D. Dolev, S. Kramer, and D. Malki. 1992. Transis: a communication subsystem for high availability. In *Digest of Papers: Twenty-Second International Symposium on Fault-Tolerant Computing (FTCS)*. IEEE, Boston, Massachusetts, USA, 76–84. https://doi.org/10.1109/FTCS.1992.243613

[2] Baber Aslam, Soyoung Park, Cliff C Zou, and Damla Turgut. 2010. Secure traffic data propagation in vehicular ad hoc networks. *International Journal of Ad Hoc and Ubiquitous Computing* 6, 1 (2010), 24–39. https://doi.org/10.1504/IJAHUC.2010.033823

[3] M. Asplund. 2015. Model-Based Membership Verification in Vehicular Platoons. In *2015 IEEE International Conference on Dependable Systems and Networks Workshops*. 125–132. https://doi.org/10.1109/DSN-W.2015.21

[4] Mikael Asplund, Jakob Lövhall, and Emilia Villani. 2017. Specification, Implementation and Verification of Dynamic Group Membership for Vehicle Coordination. In *Proceedings of the 22nd IEEE Pacific Rim International Symposium on Dependable Computing (PRDC)*. IEEE. https://doi.org/10.1109/PRDC.2017.57

[5] Mikael Asplund, Atif Manzoor, Mélanie Bouroche, Siobhàn Clarke, and Vinny Cahill. 2012. A Formal Approach to Autonomous Vehicle Coordination. In *FM 2012: Formal Methods (LNCS)*, Vol. 7436. Springer. https://doi.org/10.1007/978-3-642-32759-9_8

[6] Rasmeet S Bali, Neeraj Kumar, and Joel J.P.C. Rodrigues. 2014. Clustering in vehicular ad hoc networks: Taxonomy, challenges and solutions. *Vehicular Communications* 1, 3 (2014), 134 – 152. https://doi.org/10.1016/j.vehcom.2014.05.004

[7] N. Bissmeyer, K. Schroder, J.Y. Petit, S. Mauthofer, and K. Bayarou. 2013. Short paper: Experimental analysis of misbehavior detection and prevention in VANETs. In *Fifth IEEE Vehicular Networking Conference, VNC 2013*. IEEE, USA. http://doc.utwente.nl/89717/

[8] Felipe Boeira, Marinho P. Barcellos, Edison Pignaton de Freitas, Mikael Asplund, and Alexey Vinel. 2017. On the Impact of Sybil Attacks in Cooperative Driving Scenarios. In *in proceedings of IFIP Networking 2017 Conference and Workshops*.

[9] L. Chen and C. Englund. 2016. Cooperative Intersection Management: A Survey. *IEEE Transactions on Intelligent Transportation Systems* 17, 2 (Feb 2016), 570–586. https://doi.org/10.1109/TITS.2015.2471812

[10] Gregory V. Chockler, Idid Keidar, and Roman Vitenberg. 2001. Group communication specifications: a comprehensive study. *ACM Comput. Surv.* 33, 4 (2001), 427–469. https://doi.org/10.1145/503112.503113

[11] Leonardo de Moura and Nikolaj Bjørner. 2008. Z3: An Efficient SMT Solver. In *Tools and Algorithms for the Construction and Analysis of Systems*, C. Ramakrishnan and Jakob Rehof (Eds.). Lecture Notes in Computer Science, Vol. 4963. Springer Berlin / Heidelberg, 337–340. https://doi.org/10.1007/978-3-540-78800-3_24

[12] S. Dietzel, J. Gurtler, R. van der Heijden, and F. Kargl. 2014. Redundancy-based statistical analysis for insider attack detection in VANET aggregation schemes. In *Vehicular Networking Conference (VNC), 2014 IEEE*. 135–142. https://doi.org/10.1109/VNC.2014.7013332

[13] Daniel Ekwall. 2009. The displacement effect in cargo theft. *International Journal of Physical Distribution & Logistics Management* 39, 1 (2009), 47–62. https://doi.org/10.1108/09600030910929183 arXiv:https://doi.org/10.1108/09600030910929183

[2] www.rics.se

[14] C. Englund, L. Chen, J. Ploeg, E. Semsar-Kazerooni, A. Voronov, H. H. Bengtsson, and J. Didoff. 2016. The Grand Cooperative Driving Challenge 2016: boosting the introduction of cooperative automated vehicles. *IEEE Wireless Communications* 23, 4 (August 2016), 146–152. https://doi.org/10.1109/MWC.2016.7553038

[15] Negin Fathollahnejad, Risat Pathan, and Johan Karlsson. 2015. On the Probability of Unsafe Disagreement in Group Formation Algorithms for Vehicular Ad hoc Networks. In *Dependable Computing Conference (EDCC), 2015 Eleventh European*. IEEE, 256–267. https://doi.org/10.1109/EDCC.2015.29

[16] Jun Han, Madhumitha Harishankar, Xiao Wang, Albert Jin Chung, and Patrick Tague. 2017. Convoy: Physical Context Verification for Vehicle Platoon Admission. In *Proceedings of the 18th International Workshop on Mobile Computing Systems and Applications (HotMobile '17)*. ACM, New York, NY, USA, 73–78. https://doi.org/10.1145/3032970.3032987

[17] Attila Jaeger, Norbert Bißmeyer, Hagen Stübing, and SorinA. Huss. 2012. A Novel Framework for Efficient Mobility Data Verification in Vehicular Ad-hoc Networks. *International Journal of Intelligent Transportation Systems Research* 10, 1 (2012), 11–21. https://doi.org/10.1007/s13177-011-0038-9

[18] Amin Kharraz, William Robertson, Davide Balzarotti, Leyla Bilge, and Engin Kirda. 2015. *Cutting the Gordian Knot: A Look Under the Hood of Ransomware Attacks*. Springer International Publishing, Cham, 3–24. https://doi.org/10.1007/978-3-319-20550-2_1

[19] Chengzhe Lai, Rongxing Lu, and Dong Zheng. 2016. SPGS: a secure and privacy-preserving group setup framework for platoon-based vehicular cyber-physical systems. *Security and Communication Networks* 9, 16 (2016), 3854–3867. https://doi.org/10.1002/sec.1523

[20] M. A. Lebre, F. Le Mouel, and E. Menard. 2016. Resilient, Decentralized V2V Online Stop-Free Strategy in a Complex Roundabout. In *2016 IEEE 83rd Vehicular Technology Conference (VTC Spring)*. 1–5. https://doi.org/10.1109/VTCSpring.2016.7504449

[21] T. Leinmuller, E. Schoch, and F. Kargl. 2006. Position Verification Approaches for Vehiculer Ad Hoc Networks. *Wireless Communications, IEEE* 13, 5 (October 2006), 16–21. https://doi.org/10.1109/WC-M.2006.250353

[22] Michael W. Levin, Stephen D. Boyles, and Rahul Patel. 2016. Paradoxes of reservation-based intersection controls in traffic networks. *Transportation Research Part A: Policy and Practice* 90, Supplement C (2016), 14 – 25. https://doi.org/10.1016/j.tra.2016.05.013

[23] L. Li, J. Sun, Y. Liu, M. Sun, and J. Dong. 2018. A Formal Specification and Verification Framework for Timed Security Protocols. *IEEE Transactions on Software Engineering* 44, 8 (Aug 2018), 725–746. https://doi.org/10.1109/TSE.2017.2712621

[24] Léon Lim and Denis Conan. 2014. Partitionable group membership for Mobile Ad hoc Networks. *J. Parallel and Distrib. Comput.* 74, 8 (2014), 2708–2721. https://doi.org/10.1016/j.jpdc.2014.03.003

[25] Antonio Lima, Francisco Rocha, Marcus Völp, and Paulo Esteves-Veríssimo. 2016. Towards Safe and Secure Autonomous and Cooperative Vehicle Ecosystems. In *Proceedings of the 2Nd ACM Workshop on Cyber-Physical Systems Security and Privacy (CPS-SPC '16)*. ACM, New York, NY, USA, 59–70. https://doi.org/10.1145/2994487.2994489

[26] N. Lyamin, A. Vinel, M. Jonsson, and J. Loo. 2014. Real-Time Detection of Denial-of-Service Attacks in IEEE 802.11p Vehicular Networks. *Communications Letters, IEEE* 18, 1 (January 2014), 110–113. https://doi.org/10.1109/LCOMM.2013.102213.132056

[27] Santa Maiti, Stephan Winter, and Lars Kulik. 2017. A conceptualization of vehicle platoons and platoon operations. *Transportation Research Part C: Emerging Technologies* 80, Supplement C (2017), 1 – 19. https://doi.org/10.1016/j.trc.2017.04.005

[28] P. Papadimitratos, L. Buttyan, T. Holczer, E. Schoch, J. Freudiger, M. Raya, Zhendong Ma, F. Kargl, A. Kung, and J.-P. Hubaux. 2008. Secure vehicular communication systems: design and architecture. *Communications Magazine, IEEE* 46, 11 (November 2008), 100–109. https://doi.org/10.1109/MCOM.2008.4689252

[29] P. Papadimitratos, G. Calandriello, J.-P. Hubaux, and Antonio Lioy. 2008. Impact of vehicular communications security on transportation safety. In *INFOCOM Workshops 2008, IEEE*. 1–6. https://doi.org/10.1109/INFOCOM.2008.4544663

[30] G. Primiero, F. Raimondi, T. Chen, and R. Nagarajan. 2017. A Proof-Theoretic Trust and Reputation Model for VANET. In *2017 IEEE European Symposium on Security and Privacy Workshops (EuroS PW)*. 146–152. https://doi.org/10.1109/EuroSPW.2017.64

[31] H. V. Ramasamy, M. Cukier, and W. H. Sanders. 2002. Formal specification and verification of a group membership protocol for an intrusion-tolerant group communication system. In *2002 Pacific Rim International Symposium on Dependable Computing, 2002. Proceedings*. 9–18. https://doi.org/10.1109/PRDC.2002.1185613

[32] Fatih Sakiz and Sevil Sen. 2017. A survey of attacks and detection mechanisms on intelligent transportation systems: VANETs and IoV. *Ad Hoc Networks* 61, Supplement C (2017), 33 – 50. https://doi.org/10.1016/j.adhoc.2017.03.006

[33] Naveen Sastry, Umesh Shankar, and David Wagner. 2003. Secure Verification of Location Claims. In *Proceedings of the 2Nd ACM Workshop on Wireless Security (WiSe '03)*. ACM, New York, NY, USA, 1–10. https://doi.org/10.1145/941311.941313

[34] A Studer, M. Luk, and A Perrig. 2007. Efficient mechanisms to provide convoy member and vehicle sequence authentication in VANETs. In *Third International Conference on Security and Privacy in Communications Networks (SecureComm)*. 422–432. https://doi.org/10.1109/SECCOM.2007.4550363

[35] Roey Tzezana. 2016. Scenarios for crime and terrorist attacks using the internet of things. *European Journal of Futures Research* 4, 1 (01 Dec 2016), 18. https://doi.org/10.1007/s40309-016-0107-z

[36] S. Ucar, S. C. Ergen, and O. Ozkasap. 2016. Security vulnerabilities of IEEE 802.11p and visible light communication based platoon. In *2016 IEEE Vehicular Networking Conference (VNC)*. 1–4. https://doi.org/10.1109/VNC.2016.7835972

[37] S. Uppoor, O. Trullols-Cruces, M. Fiore, and J. M. Barcelo-Ordinas. 2014. Generation and Analysis of a Large-Scale Urban Vehicular Mobility Dataset. *IEEE Transactions on Mobile Computing* 13, 5 (May 2014), 1061–1075. https://doi.org/10.1109/TMC.2013.27

[38] A. Vora and M. Nesterenko. 2006. Secure Location Verification Using Radio Broadcast. *IEEE Transactions on Dependable and Secure Computing* 3, 4 (Oct 2006), 377–385. https://doi.org/10.1109/TDSC.2006.57

[39] S. Yan, R. Malaney, I. Nevat, and G. W. Peters. 2016. Location Verification Systems for VANETs in Rician Fading Channels. *IEEE Transactions on Vehicular Technology* 65, 7 (July 2016), 5652–5664. https://doi.org/10.1109/TVT.2015.2453160

[40] Bo Yu, Cheng-Zhong Xu, and Bin Xiao. 2013. Detecting Sybil attacks in VANETs. *J. Parallel and Distrib. Comput.* 73, 6 (2013), 746 – 756. https://doi.org/10.1016/j.jpdc.2013.02.001