Establishing Trust Between Fully-Connected Autonomous Vehicles

Jordan Hedges
School of Computer Science
The University of Western Australia
21296484@student.uwa.edu.au

Chris McDonald*
School of Computer Science
The University of Western Australia
Chris.McDonald@uwa.edu.au

Abstract

The development of autonomous vehicles by automotive and technology companies, is an exciting and active area of research receiving much media attention. Fascination extends from the independence of these vehicles, gained by measuring their local environments using a number of sensing and imaging technologies. The widely adopted SAE International's Levels of Driving Automation define a six-level scale describing vehicle independence, with higher levels indicating more sophisticated automation. Contemporary research has recently developed vehicles rated at level-3 in which vehicles can operate autonomously but do not support all driving modes, such as high speed highway driving or low speed traffic jams and, in an emergency, require driver intervention. To attain higher levels of driving automation vehicles must be able to reason about their environment to be able to support all driving modes and to react in emergencies. To achieve this, research must extend to fully-connected vehicles, in which vehicles become more aware of their environment, at greater distances, by communicating with other vehicles and transport infrastructure.

The speed and latency communication requirements of fully-connected vehicles preclude their use of 4G and even 5G telephony networks. Instead, vehicles will communicate using localised and temporarily-formed vehicle ad-hoc networks (VANETs). Without a centralised authority, VANETs are susceptible to malicious attacks from vehicles and even roadside infrastructure, which can transmit falsified information about vehicles' motion and road conditions. Such attacks have the potential to disable individual vehicles, dramatically increase congestion, and even cause potentially fatal accidents. Past research has attempted to address this problem using roadside units (RSUs), devices which supply services such as temporary authentication or systems to prevent identified malicious actors from communicating. However, these units will incur a large cost to set up, will have a transition period before they are operational and, particularly in a large country such as Australia with remote areas, cannot offer full coverage.

To better support the detection of falsified or simply errant information, we present a decentralised system which establishes localised reputation- and trust-based networks between fully-connected vehicles. Information received by each vehicle is verified by calculations that match data from physical systems on the vehicle and the degree of consensus about the same information also received by neighbouring vehicles.

*Corresponding author

I. Introduction

Autonomous vehicles, at least in the prototype stages, have arrived. Alphabet's subsidiary Waymo [19], Uber [1] and Tesla [9] all have prototypes on the road in American cities with Waymo even offering free rides in its prototype. Government bodies and advocate groups such as the Australian Driverless Vehicle Initiative cite avoidable costs due to traffic as potentially $53.3 billion per annum by 2031 in addition to $27 billion due to crashes [7]. Industry, and trucking companies in particular, are interested in reducing their transport costs for a potential saving of US $168 billion per year for the US alone [4]. However to achieve these projected savings, greater levels of automation than those currently deployed in the prototypes are required.

The highest level of automation as described by SAE International [17], level 5, could be described as "no human required" the system is in full control and no human is required, even in an emergency. The full table of these automation levels is presented in table 1. It is generally accepted that to achieve this level of automation vehicles would need to communicate both with each other and with traffic infrastructure forming a vehicle ad-hoc network (VANET). VANETs are a specialised form of mobile ad-hoc network (MANET). As noted by Blum, Eskandarian and Hoffman [3], VANETs are characterised by "rapid but somewhat predictable topology changes, with frequent fragmentation, a small effective network diameter, and redundancy that is limited temporally and functionally"citeChallenges.

With autonomous vehicles increasingly carrying and interacting with humans there is also a need for, and interest around, the safety and robustness of autonomous vehicles. Reports, such as the "Exploring Cyber Security Policy Options in Australia" report, authored by the RAND Corporation in conjunction with the Australian National University [14] are an example of this. It includes consultation with

Level	Description
0	The full-time performance by the human driver of all aspects of the dynamic driving task, even when enhanced by warning or intervention systems.
1	The driving mode-specific execution by a driver assistance system of either steering or acceleration/deceleration using information about the driving environment and with the expectation that the human driver perform all remaining aspects of the dynamic driving task.
2	The driving mode-specific execution by one or more driver assistance systems of both steering and acceleration/deceleration using information about the driving environment and with the expectation that the human driver perform all remaining aspects of the dynamic driving task.
3	The driving mode-specific performance by an automated driving system of all aspects of the dynamic driving task with the expectation that the human driver will respond appropriately to a request to intervene.
4	The driving mode-specific performance by an automated driving system of all aspects of the dynamic driving task, even if a human driver does not respond appropriately to a request to intervene.
5	The full-time performance by an automated driving system of all aspects of the dynamic driving task under all roadway and environmental conditions that can be managed by a human driver.

Table 1: *SAE International automation levels [17]*

stakeholders as well as war-gamed cyber security examples and specifically mentions vehicles as requiring stringent standards to be protected.

This interest in safety is no surprise, as a review of possible use cases for VANETs [10] covers not only the driving of a vehicle, but routing multiple vehicles including speed matching; this is in addition to interaction with users and potentially exposing their private information in order to schedule. These “basic" VANET features are in addition to contemporary research aiming to add more complex features, with a particular focus on inter-vehicle communication to enable these features. However, with an increasing amount of data and reliance on this data comes increasing risk of, and opportunities to, attack or disrupt the system whether via false information injection or attacks against the wireless communications layer. This could be accidental, due to measurement or communication error, or malicious, against the VANET and its data, or even the vehicles' operating systems.

If the information gained from a VANET is to be safely utilised and relied upon, it must first be verified. Malicious information in the network would degrade the usefulness of the network and potentially lead to life threatening accidents. It is not enough that a VANET be secured using traditional cryptography, which is reasonably possible despite the challenges of a VANET [6]. Even though the communications themselves might be secure, a malicious actor only has to misinform its communication partners in order to create chaos. Accordingly, some measure of the objective truth for all the system information must be derived and then used to inform the workings of the network and the driving of the vehicles.

To mitigate data integrity risks previous uses of VANETs have deployed roadside units (RSUs) [11], roadside infrastructure that participate in the VANET in a variety of ways, such as by message forwarding. This will be problematic as, until the RSU infrastructure is in place, vehicles will be unable to function correctly with any system that requires them. Even once RSU deployment has begun, it may be many years before the system has adequate coverage and, in places such as remote Australia, coverage may never be provided. Even in a metropolitan area RSU deployment will “incur high cost" and “is difficult in the near future" [8]. Therefore our proposed system for deriving trust in a vehicle does not currently rely on any RSUs.

Therefore, the system proposed will require each vehicle to maintain and derive their own trust in vehicles it has contact with. This will require that each vehicle is able to process information it receives from other vehicles. This will be done using information it knows is correct from physical sensors, such as angle-of-arrival, signal strength or shared information from trusted sources, and combining these to determine the veracity of received information.

II. Method

To begin, a simulator must be selected so that the proposed model can be implemented and then tested. The simulator itself should not be a barrier to implementation, given enough time the results should be replicable in another open source simulator.

i. Simulator Selection

In order to test trust solutions and attacks against these solutions, a vehicle simulator with certain attributes was required; Various simulators were therefore reviewed and selected from against the following criteria:

- Source code access: As no simulator found implemented or permitted experimentation with trust ideas, the source code of the simulator must be accessible to implement these ideas. Open source or less restrictive licenses were the most desirable.
- Programming language: In order of preference, we sought simulators written in the

C++, Java, or C programming languages.
- Up to date: Simulators that are regularly updated or have reached a completed state were preferred in addition to being responsive to fixing bugs.
- Readability and documentation: Simulators with readable and well organised source code, in addition to being well documented were preferred.
- Network and vehicle integration: A simulator that has both network simulation and vehicle simulation in a single program and did not require separate network simulation and vehicle simulation were preferred.

The simulators examined were TraNS [16], GrooveNet [12], VANETsim [21] and autoauto [13].

- *TraNS*: TraNS is open source as preferred and licensed under the Apache license version 2. TraNS is programmed in Java, the second preference. TraNS' last update was February 3^{rd} 2009, although it is unclear if it is still being developed. TraNS is both readable and documented, source files are clearly organised and a javadoc is included. TraNS combines both ns2 and SUMo to simulate a VANET, not a single simulation as preferred.
- *GrooveNet*: GrooveNet is open source as preferred, although its license status is unclear the source code is publicly available. GrooveNet is programmed in C++ as was the first preference. GrooveNet is not up to date with the last change committed on the 12^{th} of July, 2013 with a TODO list of intended changes still visible. GrooveNet is readable and well documented, with documentation and guides available on its homepage [15]. As preferred, GrooveNet combines both network and vehicle simulation in a single, unified program.
- *VANETsim*: VANETsim is open source as preferred and licensed under the GNU GPL license, version 3. VANETsim is programmed in Java, the second preference. As of the 4^{th} of April, 2017 VANETsim is no longer updated. VANETsim is readable, with a well organised structure and has available a javadoc and doxygen output for documentation. VANETsim combines both network and vehicle simulation as preferred.
- *autoauto*: autoauto is a locally written open source simulator, released under a creative commons license. *autoauto* is programmed in C++, as preferred. *autoauto* is up to date as preferred, with its current (unreleased) version being version 1.15. *autoauto* is readable with a clear file structure and doxygen used for documentation as preferred. *autoauto* combines both vehicle simulation and network simulation as a single program as preferred.

From this list, *autoauto* was selected as it fulfilled all the preferences given, in addition to the availability and proximity of expert advice.

ii. Proposed Model

As it is expected there will not be any supporting infrastructure such as RSUs at least for some period before they are established or, possibly never in rural areas, the proposed model should not rely upon them. This choice makes the establishment of trust more difficult, the RSUs in other model generally either facilitate functions that allow easier trust establishment or act as a trusted source. As there will be no RSUs each vehicle should be able to be driven with no other vehicles supporting it; each vehicle should have its own way to evaluate the trustworthiness of other vehicles. This evaluation includes tests considering physical properties such as wireless packets' signal strength and angle-of-arrival; this means the information will always be correct.

The proposed model is that each vehicle is equipped with a device named a *TrustModule* that derives the trust placed in other vehicles based on received information. This device will sit between the vehicles' driving logic and

the reception of the packet. The *TrustModule* will also keep a history of sent information so that other vehicles can enquire about sent information and to provide a more nuanced trust rating of another vehicle.

In the *autoauto* simulator, each vehicle broadcasts an update to all other vehicles within range ten times per second; receiving vehicles record each packet's arriving signal strength and angle or arrival. This update includes the senders' position and size, vehicle type, previous, current and next turns as well as their speed, acceleration and heading. In the original implementation, with no malicious attackers, these packets were then immediately passed on to the vehicles' driving logic. With the addition of malicious attackers it is important that any information passed to the driving logic is correct, this is a logical point to place the verification logic.

Each packet will be individually assessed by tests that attempt to use physical characteristics of the received packet to corroborate their information and assigned a result as to how trustworthy each packet is. These packets will then form the history used to determine whether a vehicle is trustworthy over time. This is necessary as basing the trust in a vehicle on a single packet could lead to attacks that simply alternate malicious and non-malicious packets, some record is necessary. In order form a trust rating based on multiple packets received over time, each vehicle will also keep a history of packets they have received as proposed by Chen and Wei [5]. This history will keep the entire packet and the results of the tests performed upon it, this will allow the history to then be weighted to produce a temporal effect on the trust in old data.

iii. Assumptions

To facilitate our proposed model the following assumptions are made about the equipment of the vehicle and vehicular communication:

- Vehicles carry a GPS receiver to determine their absolute position and global time
- Vehicles can determine the signal strength and angle-of-arrival of wireless packets
- Vehicles can encrypt their point-to-point communications via a method such as by Gazdar et al. [6]
- Vehicles carry a standard wireless transmitter operating on a common transmission frequency, and are able to vary its transmission power

To improve a vehicle's resistance to false information being injected into the network, a device named a *TrustModule* is implemented to review incoming transmissions.

iv. TrustModule

The *TrustModule* device attempts to determine the trustworthiness of each packet's information. Every vehicle is equipped with its own *TrustModule* which receives and assesses whether the packet is trusted enough to be acted upon by the vehicle, i.e. passed to the driving logic. Depending on the *TrustModule*'s assessment, the driving logic may or may not receive the packet, akin to a network firewall either forwarding or dropping packets. Thus, with a negative result, the vehicle functions as if it had not received the packet at all and with a positive result the driving logic will received the packet and will function as if the *TrustModule* was not active. Thus the *TrustModule* does not directly control any aspect of the driving computation.

The *TrustModule* combines the weighted results of four tests to produce a result in the range $[0, 1]$ and accepts the result if the result is above a given number in the range $[0, 1]$ similarly to Gazdar et al. [6]. These are combined simply via multiplication of the weight with the Boolean result of a test. Given a list of Boolean packet test results r and the weights of the test results w the result for a packet p is:

$$result_p = \sum_i w_i \cdot r_i$$

then simply, for a packet acceptance threshold t_{packet} if the following is true:

$$result_p \geq t_{packet}$$

the packet is accepted. This means the packet will be passed onto the history with a positive result. Once a packet is in the history, a trust rating for the sending vehicle is updated, if this passes the required threshold the most recent received packet is passed to the driving logic.

TrustModule History

The *TrustModule* maintains a history of a configurable number of received packets in order to facilitate further processing and to report to other vehicles, if questioned, the details of a recently received packet. This also allows the final trust rating in a given vehicle to be a combination of all the stored packets in addition to the most recently received packet.

Initially, when the history is empty $trust_v$, the trust in any given vehicle v will be by default 0.5, positive and negative results of packets will then increment or decrement this rating by default by 0.1 depending on the result. This is simply to give a rough estimate to how truthful the vehicle has been previously, with no weighting given based on how recent the information is while a history is built up. Once the history is built up each individual packet is then weighted based on how recently it was received.

"Bootstrapping" that is, determining whether a vehicle is trustworthy with zero prior information, particularly when a vehicle is just beginning their journey and have no received information. This was noted to be a problem by Abdul-Rahman and Hailes [2] and leaves a vehicle open to manipulation by malicious actors when they supply their trust recommendations. Their proposed solution was to have a set of trusted nodes that could always be relied upon to provide correct information. As the proposed VANET does not include any RSUs, and intersections would likely not provide enough coverage, the authors proposed solution would not work in the proposed trust model.

To counter this, the following is proposed; when a vehicle receives a packet from another vehicle it has no history for, call this vehicle B, the receiver, A, will broadcast a request to be informed about the how trustworthy the sender is. Vehicles receiving this broadcast request will then consult their history for how trustworthy the requested vehicle is and place this into their response, call the sender of one of these responses, C. Once this response packet is received by A, if C is above the trust threshold the trust in B will then be increased. The trust in B will be increased by the trust the C places in B, weighted by the trust A places in C, see below:

$$trust_{AinB} = trust_{AinC} * trust_{CinB}$$

This, in addition to the default settings, allow a sender to be quickly classified as either trustworthy or not trustworthy before the receiving vehicle has a complete history and will reduce the period of vulnerability during which the receiving vehicle could be manipulated.

Individual packets' results can further be weighted to provide a decreased trust in older events or a threshold, such as employed by Shaikh and Alzahrani [18]. Then, given a list of weights w and a matching size history of packets results h the trust result for a given vehicle v will be:

$$trust_v = \sum_i w_i \cdot h_i$$

A decreasing trust weighting in earlier packets' results means more credence is given to a vehicle's current behaviour. These weightings will depend on how the user theorises the vehicle may be attacked, potential attackers may attempt to flush other vehicles' history with "nonsense" packets in order to increase their trust rating after launching an attack. Although the packets may themselves be correct, their purpose is to flush any wrongdoing from the history of the receiving vehicle. This may mean

history weightings that favour older packets may be useful.

The trust in another vehicle will be used to determine whether a packet is passed to the driving logic, simply if $trust_v \geq t_{vehicle}$, the trust in a vehicle v is greater than or equal to the pre-set trust threshold $t_{vehicle}$. With the default settings, $t_{vehicle} = 0.5$, this would occur at the beginning with only a single trusted packet. However, once the history is full all the packets results will have to be weighted and summed to determine whether a packet is passed to the driving history.

TrustModule Verification Methods

The following are the four tests used to determine if a packet's data is meaningful. Their true or false results are combined using either a default equal weighting, or with a user provided weighting.

OnMap

The OnMap algorithm determines whether a vehicle's claimed co-ordinates and past, current and next goal turn align. Outside of a simulation, this may be achieved by the use of GPS maps which would then be required to be updated regularly for the test to be accurate. Given a claimed position from a packet p_x and p_y and the current turns the vehicle is in between p_{t0} and p_{t1} this algorithm determines whether the claimed coordinates actually lie on a road. First, the algorithm establishes, using p_{t0} and p_{t1} the road section this forms; this gives the four corners r_{ul}, r_{ur}, r_{ll} and r_{lr} the upper left, upper right, lower left and lower right corners respectively. Then to determine if a point is within the provided point set forming the road the winding number of the point for the provided point set is calculated. The winding number is the number of times a curve makes a full counter-clockwise turn around the origin, if the curve is instead a polygon and the origin a provided point, it can be determined if the point is in the polygon. If the winding number is zero, the point is outside the polygon, otherwise it is inside. To calculate the winding number, an existing implementation by Dan Sunday [20] was used.

Then if the point is determined to be within the road claimed by the provided past and current goal turns, the algorithm will return true, otherwise it will return false.

CosineSimilarity

This algorithm determines the cosine similarity, how similar two vectors are based on their orientation and magnitude, which is bound between $[-1, 1]$. Two perpendicular vectors have a similarity of zero, and two vectors of the same magnitude but opposing directions have a similarity of -1. In a positive (x, y) region, cosine similarity is instead bound between $[0, 1]$. The specific formulation for vehicles of the general formula for vectors is given by Chen and Wei [5]. The vector is composed of the vehicle's co-ordinates and its velocity. The algorithm takes in the received packet and the threshold t for how similar vectors must be to return a positive result. To calculate the cosine similarity the algorithm requires an estimate vector to compare to the claimed vector in the packet. These vector's components are named x_e, y_e and v_e for the estimate vector and x_o, y_o and v_o for the claimed vector. An estimate vector is obtained by using the last obtained packet from the sending vehicle and calculating based on heading, speed and acceleration information. For a given packet p for x_e is equal to:

$$x_e = p_x + (p_{speed}/r) * \cos(p_{heading})$$

where r is the update rate per second. Similarly y_e is equal to:

$$y_e = p_y + (p_{speed}/r) * \sin(p_{heading})$$

Finally v_e is equal to:

$$v_e = p_{speed} + p_{acceleration}$$

v_e, the estimated velocity is the speed the car was travelling plus its acceleration at the time the packet was sent. Then, taking the estimate

and the claimed vector, the cosine similarity formula is:

$$\frac{x_e \cdot x_o + y_e \cdot y_o + v_e \cdot v_o}{\sqrt{x_e^2 + y_e^2 + v_e^2}\sqrt{x_o^2 + y_o^2 + v_o^2}}$$

If the cosine similarity is higher than the given threshold t_{cos}, then true is returned, otherwise false is returned.

SignalDistanceVerification
This algorithm attempts to verify whether the claimed distance from the receiver approximates the received signal strength sent by a standard transmitter. This algorithm requires the distance in metres from the sender, derived from the co-ordinates in the packet, d, the frequency in GHz, f, the receiver and transmitter's gain in dBm, R_{gain} and T_{gain} respectively and the received dBm, R_{dBm}. First the theoretical loss, l, that would occur for distance d and frequency f is calculated. Knowing the presumed strength of the transmission signal T_{trans} in dBm, and with a 10 dBm fade margin if $T_{trans} + T_{gain} + R_{gain} - l - 10$ is less than the receive sensitivity of the receiver, the algorithm returns false. Otherwise, the algorithm attempts to check whether R_{dBm} is accurate for the provided d, via applying the *FSPLInverse* algorithm. Provided the loss in dBm and a frequency in GHz, *FSPLInverse* will return the distance required to achieve this loss, inverting the free space path loss equation. Thus if:

$$|d - FSPLInverse(R_{dBm}, f)| \leq \epsilon$$

for some ϵ representing the error, the algorithm will return true, otherwise it will return false.

GeometryVerification
As defined by Shaikh and Alzahrani [18], this algorithm attempts to verify whether the positional information in a packet is accurate. The algorithm requires the angle the packet was received on, θ, derived from always accurate physical sensors, with a known max distance the signal could travel, d_{max} and the sender's co-ordinates, x_s and y_s. The receiver's co-ordinates, x_c and y_c are also required to be known. The algorithm then defines a region by four points, x_l, x_u, y_l and y_u based on the provided information. The values of x_l, x_u, y_l and y_u depend on θ and are given by:

$$0 < \theta \leq 90^\circ : \begin{aligned} x_l &= x_c \\ x_c &= x_c + d_{max} \cdot \cos\theta \\ y_l &= y_c \\ y_u &= y_c + d_{max} \cdot \sin\theta \end{aligned} \quad (1)$$

$$90^\circ < \theta \leq 180^\circ : \begin{aligned} x_l &= x_c + d_{max} \cdot \cos\theta \\ x_u &= x_c \\ y_l &= y_c \\ y_u &= y_c + d_{max} \cdot \sin\theta \end{aligned} \quad (2)$$

$$180^\circ < \theta \leq 270^\circ : \begin{aligned} x_l &= x_c + d_{max} \cdot \cos\theta \\ x_u &= x_c \\ y_l &= y_c + d_{max} \cdot \sin\theta \\ y_u &= y_c \end{aligned} \quad (3)$$

$$270^\circ < \theta \leq 360^\circ : \begin{aligned} x_l &= x_c \\ x_u &= x_c + d_{max} \cdot \cos\theta \\ y_l &= y_c + d_{max} \cdot \sin\theta \\ y_u &= y_c \end{aligned} \quad (4)$$

Finally, the algorithm checks whether sender has provided correct information by checking whether the following inequalities are satisfied:

$$x_l - \xi \leq x_s \leq x_u + \xi$$

and

$$y_l - \xi \leq x_s \leq y_u + \xi$$

where ξ represents the error, returning true if they are satisfied, false otherwise. This can be user defined and is currently defined as one metre. As also outlined by Shaikh and Alzahrani [18], when the region formed by the

above calculations is tested with claimed position, the algorithm is not guaranteed to produce correct results for all points in the region. As noted by the authors, the percentage of the region from which correct results can be derived ranges from 84% to 100% depending on θ. The region percentage derived by Shaikh and Alzahrani [18] is defined as:

$$FLDA = \left[1 - \frac{d_{max}\cos\theta \cdot d_{max}\sin\theta}{\pi \cdot (d_{max})^2}\right] 100$$

Alone, this algorithm only determines if the co-ordinates could potentially have come from a large region indicated by the angle θ, therefore it is only effective in combination with the other algorithms.

TrustModule Consultation

In addition to the tests performed by a single vehicle whenever it receives a packet, it will also request all nearby vehicles' record of this same packet. In order to choose the correct packet, the requesting vehicle specifies the position from where the packet was claimed to have been sent, and the identity of the sender. As there may be multiple packets sent from this location, i.e. from a stopped vehicle, the most recent packet from the history is chosen. As the request for the packet is sent before any more packets can be transmitted by the sender, choosing the most recent packet from the history will always result in the correct packet being chosen. Once this is returned by all vehicles that have received the requested packet, the requesting vehicle can then verify whether the information in the two packets is in agreement. This involves computing the distance from the sender, d_1 and d_2 via received signal strength, assuming standard components using *FSPLInverse*. Then taking d_1 and d_2 and received angles θ_1 and θ_2 the following is true:

$$x_{original} = d_1 * \cos(\theta_1)$$
$$y_{original} = d_1 * \sin(\theta_1)$$
$$x_{response} = d_2 * \cos(\theta_2)$$

and

$$y_{response} = d_2 * \sin(\theta_2)$$

forming the coordinates the received packet was believed to have been sent from by the two different vehicles. These are simply compared to see if the vehicles agree to within some ϵ to return a Boolean result as below:

$$result = |x_{original} - x_{response}| \leq \epsilon \;\&\&\; |y_{original} - y_{response}| \leq \epsilon$$

The requesting vehicle then uses this result to add to the trust rating of the requested packet. If true, and the responder is trusted, the trust rating in the packet is increased by the weight given to consultation $w_{consultation}$ multiplied by the trust in the responder, $t_{responder}$, as below:

$$t_{packet} += w_{consultation} \cdot t_{responder}$$

To avoid a feedback loop, requests are only made about the packets that are original data packets, not control packets. Control packets are packets such as packet requests and packet replies.

This consultation occurs after the vehicle has made a decision in regards to the truth of the packet, i.e. the packet has either been passed to the driving logic or rejected. This is so not to delay the vehicles' driving and potentially waiting for a packet that may never arrive. Thus the consultation increasing or decreasing the rating of the packet will only affect enquiries about the packet, requests for the reputation of the sender and the overall trust put in a vehicle.

III. Results

In order to test the trust model, various attacks against the information in the network were implemented; these are termed *malicious strategies*. Each strategy affects how a malicious vehicle will attack the information in the network. *MaliciousLarge*, for example, misinforms the receiver about their position by 10 metres in a random direction.

i. Experiment 1

The experiment will takes place on a simple straight line two lane road 360 metres long, traffic is able to enter ends of the road and travels in the opposite direction to the other lane.

Experiment Setup

In this experiment the *MaliciousLarge* strategy is tested. Under this strategy the receiver is misinformed of the malicious vehicle's position by 10 metres in a random direction. The chance of creating a malicious vehicle instead of a normal vehicle is 20%, giving a non-malicious to malicious vehicle ratio of 4:1. In order to compare how effective the trust model is the number of malicious packets, number of malicious packets caught, number of non-malicious misclassified packets, and total number of packets is recorded in the reports. The percentages of malicious packets caught to those sent can then be compared across experiments, as can the percentage of misclassified non-malicious packets. The results from the described experiment are presented below.

Experimental Results

Figure 1 presents four samples of the results of the experiment beginning at 5 seconds into the simulation, then 50 seconds, 100 seconds, 150 seconds and 195 seconds.

These results show that the given tests are able to detect the *MaliciousLarge* strategy with a good degree of accuracy, ranging from 60% to 77.3%, although the initial sample at 5 seconds into the simulation has a small sample size. In addition non-malicious packets are misclassified at a low rate, ranging from 0% at 5 seconds into the simulation to 1.12% at 195 seconds into the simulation. Further tuning of the *CosineSimilarity* test parameters may produce better malicious packet detection results. The raw results are given in Figure 1 and the percentages for these results are given in Figure 2.

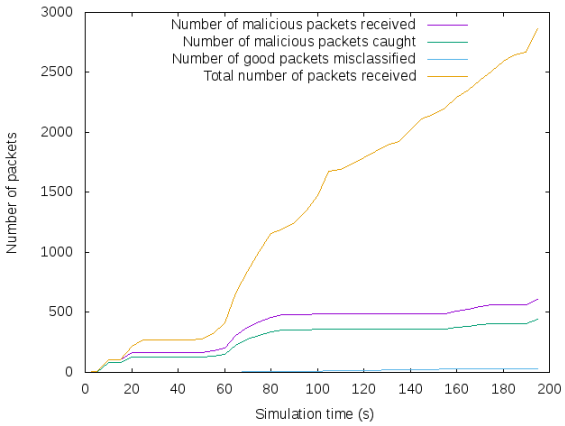

Figure 1: *Experiment 1: Graph of packet classification results*

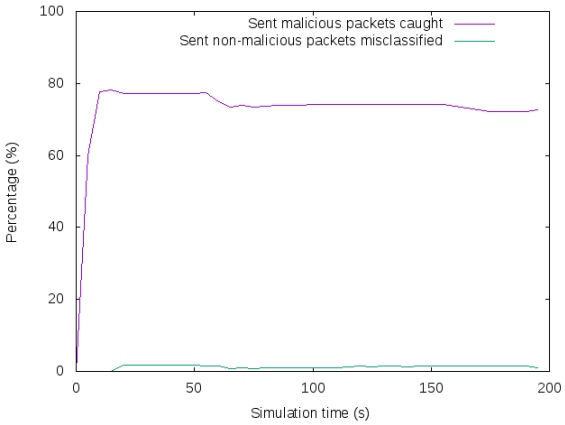

Figure 2: *Experiment 1: Graph of packet classification percentages*

ii. Experiment 2

In this experiment the effect of varying how large the misinformation is in the *MaliciousLarge* strategy on the detection rate of the malicious packets is examined. To do this, the same topology file as experiment 1 is invoked, but the size of the misinformation in the *MaliciousLarge* strategy is changed.

Experiment Setup

For this experiment, the value of how large the misinformation is by the *MaliciousLarge* strategy will be adjusted. The proposed

values of the size of the misinformation is 1 (metre), 3, 5, 10 (default value), 20 and 40. The varying values will then be compared on their size against their detection rate, the ratio of captured malicious packets to transmitted malicious packets. For this value the last report at 195 seconds into the simulation is taken.

Experimental Results
Below is a table listing the varying misinformation values and how they had performed after 195 seconds of simulation in the following categories: malicious packets sent, malicious packets caught and the percentage of these values.

These results show that varying the parameters of the malicious attack against the network does affect the detection rate, however the change has to be quite large for the change in detection rate to be noticeable. Varying the size of the misinformation in the malicious packets from 1-20 metres does not significantly vary malicious packet detection rates. However, the variation from 20 to 40 metres gives a malicious packet detection rate increase of approximately 8.5%. This indicates that the parameters in the packet tests require additional tuning, increasing the *CosineSimilarity* threshold would make this test more sensitive to position changes for example. However, this would then also likely increase the non-malicious packet misclassification rate, sudden changes in acceleration such as at traffic lights or through corners would then be more likely to trigger this test. Alternatively, this may indicate that the weighting of the packet test results may need to be changed, as the *CosineSimilarity* test is not sensitive to even large changes until a certain threshold, its results may be considered less trustworthy.

These results are given in below graphical form in Figure 3.

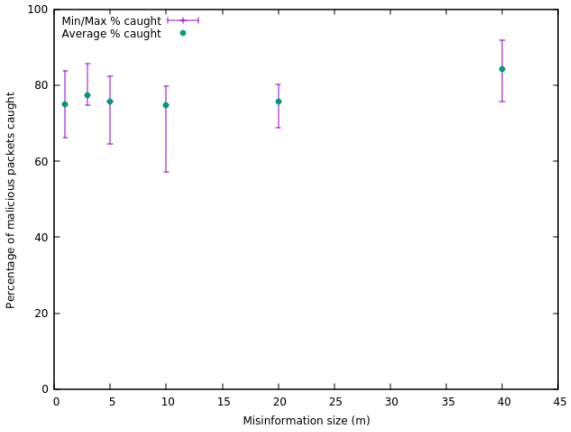

Figure 3: *Experiment 2: Graph of malicious packets caught percentage varying misinformation size*

IV. Discussion

i. Summary

The original motivation for implemented trust model was to determine whether it would be useful for determining trustworthy vehicles in a VANET based on the information they have sent. This is important because autonomous vehicles still require improvements in order to achieve the highest levels of automation as defined by SAE International [17]. To achieve level 5, the highest of the six levels of automation defined by SAE International, autonomous vehicles must effectively be able to plan their routes and interact with the environment, including infrastructure and other vehicles, in order to operate. Generally, VANETs include RSUs in order to facilitate network features, such as forwarding of packets, that may not be guaranteed by a VANET. However due to the predicted difficulties of implementing RSUs we decided not to use RSU in our trust model, this makes the implementation of the trust model more difficult as there is no implicitly trustworthy sources of information, all information received must be tested. The problem is then to determining trustworthy vehicles in a VANET without RSUs based on the information they have sent.

In summary, based on previous research and ideas, a trust model is implemented to determine which vehicles and what information in the network is trustworthy. Based on previous work that suggested RSUs would be infeasible in the near future, and potentially never in remote regions, the system has been designed to work without RSUs. This makes establishing trust more difficult as there is no implicitly trustworthy nodes in the network, compared with traditional Internet infrastructure which has the implicitly trustworthy certificate authorities.

In order to determine whether the information transmitted wirelessly by a vehicle is trustworthy, each packet is subject to four test predicates. The four tests, *CosineSimilarity*, *OnRoad*, *SignalDistanceVerification* and *GeometryVerification* all attempt to tie the claimed information from a vehicle to some observed (and implicitly correct) physical data, such as signal strength for *SignalDistanceVerification*. These tests are then combined to give a rating for each individual packet which is passed into the vehicles' configurable sized history. Prior to a vehicle filling its history the trust in another vehicle is simply determined by the number of malicious and non-malicious packets it has received from the sender, no weighting is applied. However, once the packet history is full, the trust in a sender is a weighted sum of the packets contained in the history, allowing a temporal factor to be added to the trust in other vehicles. Once a packet has been received, a vehicle also consults with nearby vehicles to see if they have received the packet, and if so, to return a copy of the information contained in the packet. If the two vehicles agree on the information contained in the packet, and it is physically consistent, the trust in the packet is increased. In addition, when a vehicle receives a packet from a previously unknown sender it may request the trust in the sender from other nearby vehicles. This is then weighted by the trust in the respondent. This technique helps to facilitate the bootstrapping of trust values when a vehicle initially joins a VANET.

In order to test the trust model described in this paper, it was implemented as a series of dynamically loadable modules for the *autoauto* vehicle and network simulator. This was chosen due to its open source nature and being a combined vehicle and network simulator. In order to mimic potential attacks against the network, a group of malicious strategies were implemented, these effectively detailing how the information in the network is to be attacked. The strategies *MaliciousLarge* and *MaliciousSmall* deliberately misinform about the position of their owning vehicle in large and small quantities, respectively. *MaliciousPower* transmits information at a different provided strength than to what is assumed to be standard. *MaliciousShadow* responds to any packets by claiming to be a vehicle length in front of the sender. These strategies were then tested in a series of experiments to determine the effectiveness of the packet tests and trust model.

Results from a number of other experiments, not reported here, indicate that trust models are well suited for use in VANETs. The implemented tests are effective in detecting the implemented malicious attacks and also have a low degree of misclassification of non-malicious packets. However, the detection rate of malicious packets is also related to the parameters employed in the simulation, varying these parameters dynamically may increase detection rate, this is discussed in the Further Work section. Ideally, the misclassification rate of non-malicious packets should be zero, adjusting of test parameters may alleviate this problem.

In addition the results from experiment 2 indicate that the specific parameters the malicious vehicles attack the network with affect the detection rate. Comparison of the results of the packet tests to determine how large the misinformation used by detected packets are could be used to inform dynamic variation of the packet test parameters, this is discussed in the Further Work section.

A final experiment has shown that mali-

cious vehicles attacking the network increases the overhead of the network, specifically the amount of data that must be processed by each vehicle. When running a simulation on a desktop computer execution time of the tests is unlikely to be problematic for the small percentage of packets that are actually malicious. However, when running on the much weaker hardware likely to be present in vehicles, any extra tests that have to be used because extra malicious packets were received could delay computation on safety critical data such as sensors.

Trust models are effective in the context of VANETs, their decentralised nature allows each vehicle to form their own view on how trustworthy other vehicles are. In addition, trust models also deal well with the ad-hoc nature of the network, they simply test whatever information they receive; they do not rely on any strict timings for sent or received messages, it is never assumed the packet may reach its target or anyone at all. However, each vehicle does need to be equipped with the *TrustModule* itself in addition to other devices we assumed the vehicles were equipped with such as a GPS and a device to detect angle-of-arrival of information. In addition to this, it is assumed that each vehicle is using a wireless device with the same characteristics, such as signal strength and signal frequency. Standardisation of hardware, especially of hardware produced by private companies may be difficult unless required by law. We have shown that the given packet tests are effective at using data from physical sensors in addition to the information received from other vehicles to determine whether the information is consistent. These tests may also have to be added or removed from the *TrustModule* as new attacks or sources of error are found or eliminated, the mechanisms for how the software ran by the *TrustModule* is to be updated is unclear.

ii. Further Work

The implemented model offers lots of opportunities for work to be extended. In particular the parameters, the threshold values, history size and weightings applied to tests and packet history could conform to an adaptive strategy. Instead of being set at the beginning of a simulation, or when a vehicle starts a journey, the values could adjust based on responses other vehicles.

Test weightings could also be adjusted based on the environment, signal strength based methods may be less useful in an urban environment which will cause refraction, diffraction or reflection. Altering packet and vehicle trust thresholds may be useful when there is a larger than expected untrustworthy to trustworthy vehicle ratio and trusting another vehicle must be subject to more stringent conditions. In addition, the vehicles could also adapt to traffic conditions, e.g. traffic jams versus empty roads, or physical qualities of the road network such as the road layouts. These values could be adjusted based on the comparison of the ratio of trustworthy to untrustworthy vehicles, if this varies from historical or expected values the test weights could be changed. With dynamic weightings it would be expected that malicious packets would be more likely to be identified and non-malicious packets less likely to be misclassified. Alternatively AI methods could be used to search for optimal configurations of parameters in advance.

In addition further trust tests and malicious strategies could be developed. Currently, there is only basic collaboration between vehicles for both trust testing and malicious attacks, both of these could be improved with greater collaboration. Collaborative malicious attacks could use groups of malicious vehicles to collaboratively attack or misinform the group of non-malicious vehicles nearby. This data could then be used to tailor attacks so the false information seems to be physically consistent, e.g. with false coordinates and a known distance to target, the

power of transmission could be set to produce internally consistent received signal strengths. Non-malicious vehicles could themselves form trusted groups but this does not stop the issue of a potential well behaved "spy" informing malicious vehicles about the non-malicious vehicles. Further trust tests would then need to be developed to counter the increasing sophistication of attacks. The conditions to execute this type of attack and how they might reasonably be countered has been an extensive field of study in MANETs, so similar principles may be applicable in a VANET context.

Currently, the network simulation uses the free space path loss model which, by definition, expects free space in between the the sender and receiver. Therefore, the given results are the "best case" scenario for, in reality it is expected that both the network tests and the network itself would perform worse due to reflection, refraction and diffraction and other physical phenomena of signals. Further work could introduce this to the network simulation and then observe the (likely negative) effect this would have on the performance of the VANET. Further tests that compensate for these challenges would then need to be implemented.

V. Acknowledgments

We are grateful to Trent Reid for a number of early discussions on simulator design and usability, which guided the implementation of our own *autoauto* simulator.

References

[1] ABC News. Uber launches driverless car service in landmark US trial. www.abc.net.au/news/2016-09-14/uber-launches-groundbreaking-driverless-car-service-in-us/7845820, September 2016.

[2] Abdul-Rahman, A., and Hailes, S. Supporting trust in virtual communities. In *Proceedings of the 33rd Annual Hawaii International Conference on System Sciences* (Jan 2000), pp. 1–9 vol.1.

[3] Blum, J. J., Eskandarian, A., and Hoffman, L. J. Challenges of intervehicle ad hoc networks. *IEEE Transactions on Intelligent Transportation Systems 5*, 4 (Dec 2004), 347–351.

[4] Catapult. Shipping and Autonomous Vehicles: The Future of Logistics Technology. www.gocatapult.com/blog/shipping-and-autonomous-vehicles-the-future-of-logistics-technology/, November 2016.

[5] Chen, Y. M., and Wei, Y. C. A beacon-based trust management system for enhancing user centric location privacy in vanets. *Journal of Communications and Networks 15*, 2 (April 2013), 153–163.

[6] Gazdar, T., Benslimane, A., Rachedi, A., and Belghith, A. A trust-based architecture for managing certificates in vehicular ad hoc networks. In *2012 International Conference on Communications and Information Technology (ICCIT)* (June 2012), pp. 180–185.

[7] Haratsis, B. Economics Impacts of Automated Vehicles on Jobs and Investment. advi.org.au/wp-content/uploads/2016/09/ADVI-Economic-Position-Paper-30-09-2016-3.pdf, September 2016.

[8] Kim, D., Velasco, Y., Wang, W., Uma, R., Hussain, R., and Lee, S. A new comprehensive rsu installation strategy for cost-efficient vanet deployment. *IEEE Transactions on Vehicular Technology PP*, 99 (2016), 1–1.

[9] Krok, A. Tesla is now testing autonomous vehicles on public California roads.

https://www.cnet.com/roadshow/news/tesla-is-now-testing-autonomous-vehicles-on-public-california-roads/, February 2017.

[10] Lèbre, M.-A., Mouël, F. L., Ménard, E., Dillschneider, J., and Denis, R. Vanet applications: Hot use cases. *arXiv preprint arXiv:1407.4088* (2014).

[11] Lee, E.-K., Gerla, M., Pau, G., Lee, U., and Lim, J.-H. Internet of vehicles: From intelligent grid to autonomous cars and vehicular fogs. *International Journal of Distributed Sensor Networks 12*, 9 (2016), 1550147716665500.

[12] Mangharam, R., Weller, D., Rajkumar, R., Mudalige, P., and Bai, F. Groovenet: A hybrid simulator for vehicle-to-vehicle networks. In *2006 3rd Annual International Conference on Mobile and Ubiquitous Systems - Workshops* (July 2006), pp. 1–8.

[13] McDonald, C. autoauto Home Page. staffhome.ecm.uwa.edu.au/00014979/autoauto/, 2017. Accessed: 29/05/2017.

[14] Mikolic-Torreira, I., Snyder, D., Price, M., Shlapak, D., Beaghley, S., Bishop, M., Harting, S., Oberholtzer, J., Pettyjohn, S., Weinbaum, C., and Westerman, E. Exploring cyber security policy options in australia, August 2017.

[15] mLab. GrooveNet. mlab.seas.upenn.edu/projectsites/groovenet/.

[16] Piorkowski, M., Raya, M., Lugo, A., Papadimitratos, P., Grossglauser, M., and Hubaux, J.-P. TraNS: Realistic Joint Traffic and Network Simulator for VANETs. *ACM SIGMOBILE Mobile Computing and Communications Review 12*, 1 (2008), 31–33.

[17] SAE International. U.S. Department of Transportation's New Policy on Automated Vehicles Adopts SAE International's Levels of Automation for Defining Driving Automation in On-Road Motor Vehicles. https://trustedcomputinggroup.org/tpm-main-specification/. Accessed: 2017-04-27.

[18] Shaikh, R. A., and Alzahrani, A. S. Intrusion-aware trust model for vehicular ad hoc networks. *Security and Communication Networks 7*, 11 (2014), 1652–1669.

[19] Stewart, J. Google's Finally Offering Rides in Its Self-Driving Minivans. https://www.wired.com/2017/04/googles-finally-offering-rides-self-driving-minivans/, April 2017.

[20] Sunday, D. Inclusion of a Point in a Polygon. geomalgorithms.com/a03-_inclusion.html, December 2015.

[21] Tomandl, A., Herrmann, D., Fuchs, K. P., Federrath, H., and Scheuer, F. Vanetsim: An open source simulator for security and privacy concepts in vanets. In *2014 International Conference on High Performance Computing Simulation (HPCS)* (July 2014), pp. 543–550.