TruMan: Trust Management for Vehicular Networks

Renan Greca and Luiz Carlos Pessoa Albini
Department of Informatics – Federal University of Paraná (UFPR) – Curitiba, Brazil
Email: rdmgreca@inf.ufpr.br, albini@inf.ufpr.br

***Abstract*—By integrating processors and wireless communication units into vehicles, it is possible to create a vehicular ad-hoc network (VANET), in which cars share data amongst themselves in order to cooperate and make roads safer and more efficient. A decentralized ad-hoc solution, which does not rely on previously existing infrastructure, Internet connection or server availability, is preferred so the message delivery latency is as short as possible in the case of life-critical situations. However, as it is the case with most new technologies, VANETs will be a prime target for attacks performed by malicious users, who may benefit from affecting traffic conditions. In order to avoid such attacks, one important feature for vehicular networks is trust management, which allows nodes to filter incoming messages according to previously established trust values assigned to other nodes. To generate these trust values, nodes use information acquired from past interactions. Nodes which frequently share false or irrelevant data must have lower trust values than the ones which appear to be reliable. This work proposes TruMan, a trust management model in the context of daily commutes, utilizing the Working Day Movement Model as a basis for node mobility. The results prove to be accurate, detecting nearly all malicious nodes with very few false positives when they constitute up to 50% of the network. The model is also very efficient thanks to the low complexity of the algorithm constituting the trust model.**

I. Introduction

Within the next few years, a substantial share of new vehicles will come equipped with networking features [1]. These features will allow vehicles to quickly share data with other nearby devices and can be useful tools to reduce traffic and the risk of accidents. Over one million people lose their lives to traffic accidents every year [2], so solutions to improve road safety are crucial for modern life. By quickly sharing data with neighboring vehicles without the need of an Internet connection, smart vehicles can alert drivers of important road conditions [3], while autonomous vehicles can synchronize their movements to maximize traffic throughput [4].

The communication standard for vehicular communication is the IEEE 802.11p or Wireless Access in Vehicular Environments (WAVE) [5]. It describes two types of nodes for vehicular networks: on-board units (OBUs) and road-side units (RSUs). Communication between two OBUs is called vehicle-to-vehicle (V2V) communication, while communication between an OBU and an RSU is called vehicle-to-infrastructure (V2I) communication. This study focuses only on V2V scenarios, and therefore, any references to Vehicular ad-hoc networks (VANETs) and their nodes refer exclusively to vehicles with on-board units.

As expected for new technologies, vehicular communications can become an appealing target for malicious users and attackers. Some issues that could be exploited in such network include: vehicles with faulty sensors [6]; vehicles broadcasting false data [7]; a flood of false data to generate a distributed denial of service (DDoS) scenario or to divert traffic [8]; eavesdropping on other vehicles' communications, signal jamming or stalking [6].

Each of these problems requires specific solutions, although there are ways of making the network safer in general. One way is taking advantage of the concept of trust between network members. By having nodes remembering previous interactions with one another, it is possible for them to build trust relationships and avoid those attacks that involve the spread of false data. Trust solutions for VANETs are generally classified into data-oriented, emphasizing the message contents, or entity-based, emphasizing message senders.

This work proposes a new and improved entity-based trust model to compute the trust between any pair of nodes in a vehicular network. Using the proposed trust model, nodes in a vehicular network are able to identify which other nodes are likely malicious. Since the network is dynamic, nodes acquire more knowledge as time passes and the results of the algorithm become more precise, taking advantage of social properties of VANETs [9] [10] to build strong relationships between frequently connected nodes. Simulations demonstrate that nodes are able to correctly identify more than 95% of nodes in the network (malicious or not). The algorithm correctness depends upon the velocity of the nodes, the frequency of the information exchanged between them and the running time, i.e. the longevity, of the algorithm.

The remainder of this paper is organized as follows. Section II details the proposed trust model; Section III contains the simulation results which demonstrate the usefulness of the model; Section IV presents the previously published trust models for vehicular networks and compares them with TruMan; and Section V has the conclusion and future work directions.

II. TruMan

The objective of the TruMan trust model is to allow nodes to infer whether or not other nodes in the network are malicious. The algorithm that dictates the trust model runs continuously, with iterations happening in preset intervals. In every iteration, a node collects information from its neighbors and runs a combination of algorithms to detect malicious nodes in the known network. This information is maintained in a directed graph $T = (V, E)$, in which V is the set of known vertices and E is the set of known trust relationships between pairs $u, v \in V$. Graph T is known as the *trust graph*.

Each edge $(u, v) \in E$ stores a value between 0 and 1 which represents the degree of trust u has for v. This value is called the *opinion* of u about v. At the start of the execution, edges have value 0.5; this value increases when nodes have positive interactions and decreases otherwise. A threshold $0 < h < 1$ is used to define the minimum weight for a positive edge, meaning that the origin node trusts the destination node.

After collecting information from other nodes, TruMan performs two steps: (i) it divides the network graph into strongly connected components using Tarjan's algorithm; (ii) it uses a graph coloring algorithm as a heuristic to determine which nodes to trust or not.

The detailed descriptions of both algorithms are below, followed by the complete process of each iteration of the Truman trust model.

A. Tarjan's strongly connected components algorithm

The use of Tarjan's strongly connected components algorithm [11] is an important aspect of TruMan's efficiency. This allows a large graph to be abstracted into a smaller one, which therefore reduces the input for further steps. Given a directed graph $T = (V, E)$, a strongly connected component is defined as a group of nodes in which, for any pair of nodes $u, v \in V$, there is a path from u to v and a path from v to u. For the purposes of trust management, this definition is extended to accept only paths of edges with weight above a predetermined threshold h. Every node of graph T must belong to a component.

The number of components is, at most, $|V|$: in a worst-case scenario, each node is placed into its own component. The complexity is $O(|V| + |E|)$ for a graph $T = (V, E)$.

From the output of Tarjan's algorithm, an undirected component graph $C = (V', E')$ is formed. Each $v' \in V'$ is the abstraction of one component identified by Tarjan's algorithm, while the edges $e' \in E'$ are edges from T between nodes that do not belong in the same component (e.g. if u and v are separated into u' and v', $e = (u, v)$ becomes $e' = (u', v')$.

B. Graph coloring with minimum colors

The algorithm proposed in [12] is an efficient approach to graph coloring. Graph coloring is one of the possible heuristics used to detect malicious nodes after the generation of the component graph using Tarjan's algorithm. Out of the tested heuristics, it presents the best results, so it has been chosen as the heuristic for the trust model.

It has been mathematically proven that any planar graph can be colored with at most four colors [13], but discovering the smallest number of colors necessary to color an arbitrary graph (called the chromatic number of the graph) is an NP-hard problem [14]. In [12], the authors propose to color a graph using the minimum possible amount of colors. Although they do not prove that their algorithm always uses the smallest possible amount of colors, the output is always a correct coloration and the algorithm is nevertheless efficient. The complexity of the algorithm is $O(|E'|)$ for a graph $C = (V', E')$. As a comparison, the classic DSATUR algorithm for graph coloring has complexity $O(|V'|^2)$ [15].

C. The TruMan algorithm

TruMan is based on the MaNI algorithm [16], which suggested the use of Tarjan's algorithm and the graph coloring algorithm. However, MaNI was developed for static networks such as social networks, and is executed by an external supervising agent (i.e. outside of the network), making it unsuitable for a vehicular network.

In order to work with dynamic networks, the TruMan algorithm runs iterations at predetermined intervals. Furthermore, the algorithm runs in a decentralized fashion, meaning each node in the network runs its own instance of the algorithm. Each node starts knowing information only about itself and maintains its own abstraction of the network surrounding it. Every node u stores a representation of the network in the form of a static, connected and directed trust graph $T = (V, E)$, in which V is the set of nodes node u is aware of and E is the set of trust relationships (opinions) u knows between members of V. Since each node has its own network representation and it changes over time, there is a $T_i^u = (V_i^u, E_i^u)$ for every node u and iteration i.

At first, the node collects and organizes information. A prerequisite of this step is a test that correctly classifies a neighboring node as benign or malicious. Testing the correctness of neighboring nodes is a problem in and of itself, which is beyond the scope of this paper, but studies on this topic can be found on [7], [17], [18].

Every time a neighboring node v is tested as benign, the value of $u \to v$ increases and the trust graph T_{i-1}^v is merged into u's trust graph. After this, a new T_i^u is formed, which is used for the remaining steps.

After the collection of data, T_i^u is separated into strongly connected components using Tarjan's algorithm [11]. For each node in a component, there is a path formed by edges of weight higher than the threshold h to each other node in the same component. In other words, within a single component, all nodes trust one another directly or indirectly; nodes that do not satisfy this condition are separated into different components. Each of these components becomes a node of a component graph $C_i^u = (V_i'^u, E_i'^u)$.

The creation of C_i^u simplifies the remaining computation. Since each vertex $v' \in V_i'^u$ is a component of T_i^u in which all nodes trust each other, for the purposes of identifying malicious nodes, all nodes within each of those components can be treated as the same. They can either be benign nodes which legitimately trust one another, or malicious nodes colluding with each other. After the formation of C_i^u, a heuristic is used to classify the nodes as benign or malicious.

The coloring heuristic is used to classify nodes, which uses the algorithm described in II-B [12], although other heuristics may be considered. After running the graph coloring algorithm with graph C_i^u as input, the color whose nodes in C_i^u represent the most nodes in T_i^u is classified as correct, and all others are classified as malicious. Once this information from C_i^u is brought back to graph T_i^u, it is trivial to label the nodes in T_i^u as either benign or malicious based on the classifications of their components.

The complexity of the whole algorithm can be calculated by adding the most costly operations involved. As discussed above, Tarjan's algorithm has a complexity of $O(|V|+|E|)$ (for the trust graph T), while the graph coloring algorithm has a complexity of $O(|E'|)$ (for the component graph C). The most costly part of the algorithm is the graph merge operation that happens between trusted nodes. The complexity of the graph merge algorithm is $O(|E|)$ for each neighbor a node has in an iteration; this number is at most $|V|$. The total complexity of TruMan is, therefore, $O(|V|+|E|)+O(|E'|)+O(|V|\times|E|)$. Since $|E'| \leq |E|$ and $|V|+|E| \leq |V|\times|E|$, the complexity can be simplified to $O(|V|\times|E|)$.

In summary, every node u runs the following steps in each iteration to detect malicious nodes in the network:

1) Node u checks which are its neighbors. New discovered nodes and new formed edges are added to T_i^u. Edges are created with weight 0.5.
2) Node u tests all its neighbors to discover which ones can be directly trusted or not. New trust values are computed for the edges using the average between the previous value and either 1 (if the neighbor is trustworthy) or 0 (otherwise).
3) If a neighbor v is trustworthy, u merges T_{i-1}^v into T_i^u.
4) Tarjan's algorithm is executed to identify the strongly connected components of T_i^u, resulting in a component graph C_i^u.
5) The graph coloring algorithm is executed on C_i^u and nodes are classified as benign or malicious.

III. Results

In order to test the TruMan trust model, simulations were made using an implementation of the algorithm in Python. To generate the input graphs with node mobility, the ONE simulator [19] was used in conjunction with the Working Day Movement Model [20], which provides a strong similarity with vehicle movement in real life. Snapshots of the network were taken every 10 simulated seconds, and these snapshots were used as input for the algorithm. Malicious nodes misbehave by randomizing the values of their neighbors' opinions.

TABLE I: Simulation parameters

Parameter	**Value**
Duration	86400 seconds
Work day length	28800 seconds
Std. dev. departure time	7200 seconds
Node velocity	7 10m/s
Simulation area	approximately 14km^2
Number of nodes	150 (WDMM) + 10 (random)

Most of the parameters for the simulator were taken from the article detailing the Working Day Movement Model [20]. Different parameters are shown in I. Since this simulation is for vehicles instead of pedestrians, there are no buses in the model and every node is guaranteed to own a vehicle and travel by car. The parameters regarding offices, meeting spots and shopping were kept intact. Few nodes move randomly to simulate vehicles that do not follow daily patterns.

A. *Network Density*

The communication range varies from 10m to 50m, to illustrate the impact of different network densities. Network density (δ) is a value which abstracts the volume and frequency of connections by estimating how much of the environment is covered by the network. For TruMan, higher densities yield better results, since nodes can construct and update their models of the network faster (this is demonstrated in III-B). It is calculated using the transmission range (ρ), the amount of nodes (η), and the total area (α).

The coverage of a single node is the circumference around it formed by the transmission radius. This is divided by two to compensate for overlapping, then multiplied by the number of nodes to estimate the maximum coverage area. Finally, the value is divided by the total environment area. The network density formula is: $\delta = \frac{\frac{\rho^2\pi}{2}\times\eta}{\alpha}$

Simulations shown here have densities between 0.001 ($\rho =$ 10m) and 0.04 ($\rho =$ 50m). As a comparison, the density of the city of São Paulo (Brazil) was calculated as 2.24 with $\rho =$ 10m, a much higher value than what is necessary for a satisfactory performance of the algorithm.

B. *Simulations*

To improve readability, all figures in this section follow the same format: X axis shows the results of sequential iterations, ranging from 0 to 8639; Y axis shows a percentage of all nodes in the network, ranging from 0 to 100; blue line represents the percentage of nodes detected out of the complete network; magenta is the percentage of malicious nodes in the network (ground truth); green represents the nodes correctly identified as malicious (true positives); cyan represents the undetected malicious nodes (false negatives); red represents the benign nodes incorrectly identified as malicious (false positives).

1 shows the results of simulations running with 10% of nodes acting maliciously, with communications range varying from 10m to 50m. It is possible to see how the increase in the range allows the algorithm to converge sooner, taking over 8000 iterations with 10m range and achieving solid results at just over 1000 iterations with 50m range.

2 shows the variation of results for different amounts of malicious nodes in the network. By the end of one day, the algorithm is able to detect all malicious nodes when they are up to 30% of the network. At 40%, a small part of malicious nodes are yet to be detected. At 50%, as expected, the results are inconsistent as the network is completely split between benign and malicious nodes; at this point, the network is completely compromised. The amount of malicious nodes also affects the convergence of the algorithm, since nodes do not trust information from malicious neighbors.

3 shows the execution of the algorithm over the course of 7 days. Most malicious nodes are identified by the end of the first day; in the following iterations, the algorithm finishes building the network model and sorts out remaining false negative or false positive results. After iteration 20000, the results are completely consistent.

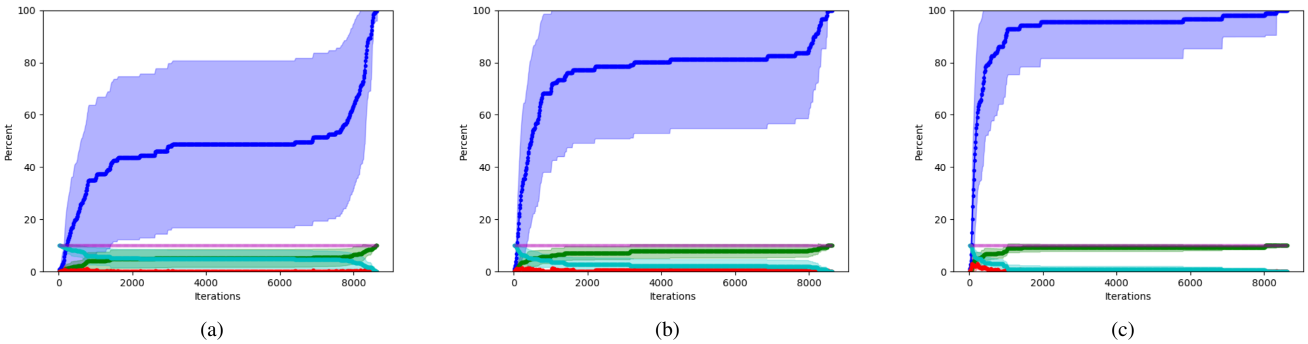

Fig. 1: Simulation with 10% malicious nodes and ρ = (a) 10m, (b) 30m or (c) 50m.

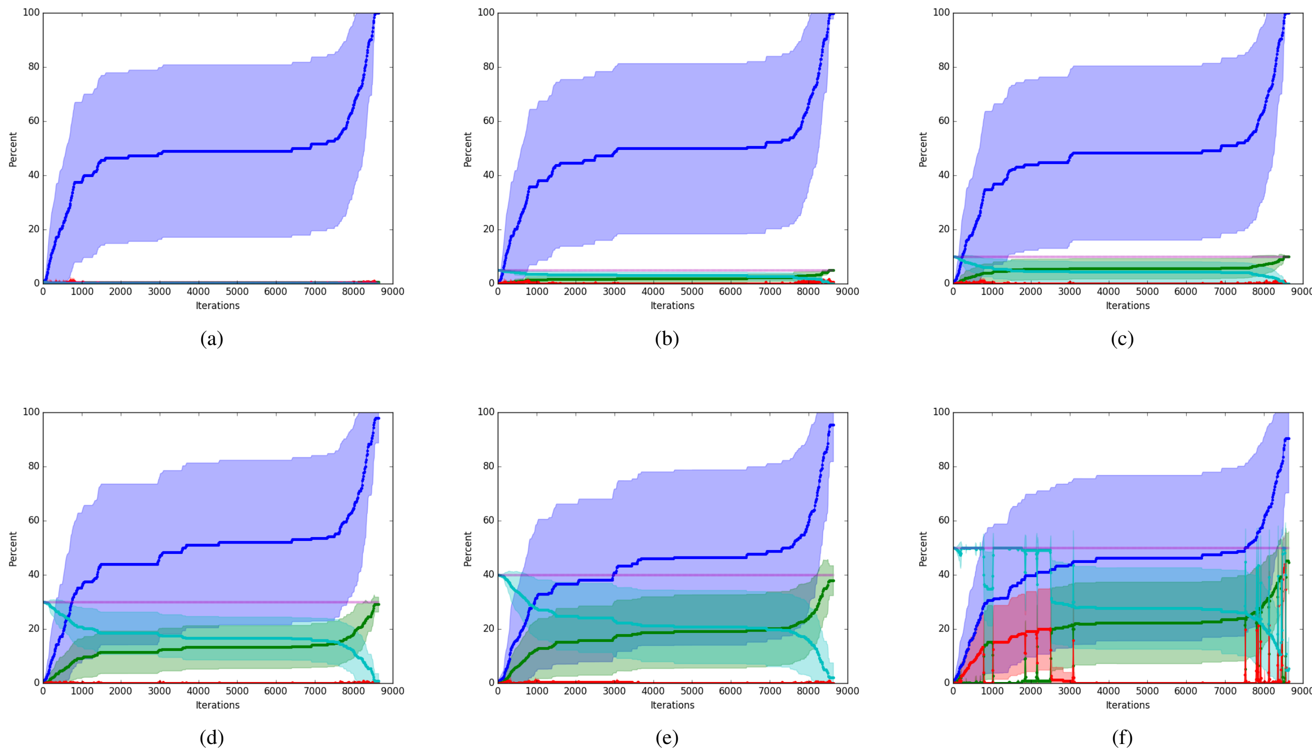

Fig. 2: Simulation with ρ = 10m and (a) 1%, (b) 5%, (c) 10%, (d) 30%, (e) 40% or (f) 50% malicious.

IV. Related work

Several models have been proposed to solve the problem of trust in vehicular networks. This analysis of related work is based on [21], which proposes eight desired properties for a trust management model for VANETs. In the first part of this section, these properties are described with an assessment of whether or not TruMan satisfies their conditions. Then, some of the most relevant models are described, considering how well they satisfy the desired properties. II shows how they compare with TruMan.

A. Properties

Decentralized trust establishment: nodes must be able to form their own trust values about other nodes. Nodes may or may not use information from other trustworthy nodes to build trust values. TruMan satisfies this as it is built from the ground up for decentralized systems.

Coping with sparsity: the model still functions when there are few nodes populating the network. The experiments using low density values demonstrate that TruMan works in reasonably sparse networks. Due to its decentralized nature, it can also work on isolated chunks of the network.

Event/task and location/time dynamics: how the model

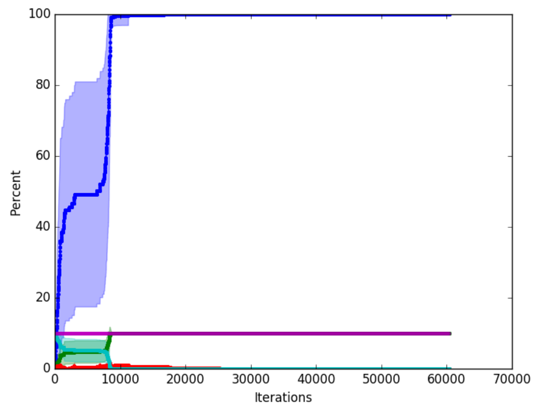

Fig. 3: 7 days scenario: 10m range and 10% malicious nodes.

reacts to different situations depending on what, where and when events happen. Although this has not been used in the simulations in this paper, TruMan can easily be extended to consider time and location as long as nodes store geolocation and timestamp data.

Scalability: the model can work on very large networks at high speeds. Due to the low complexity of the algorithms used in the model, TruMan can be highly scalable, as it does not incur substantial pressure on the vehicles' on-board units. It has also been demonstrated that iterations of the algorithm do not need to run extremely frequently in order to detect malicious nodes with high accuracy.

Integrated confidence measure: allows nodes to estimate how useful the output of the algorithm is. Since nodes using TruMan store trust values as a number between 0 and 1, this value can be used as a confidence measure of the opinion. The closer it is to 1, the higher the chance that it is accurate.

System level security: requires authentication of nodes participating in the network. This has not been considered in evaluations of TruMan. However, it can be included as a separate security model during the transmission of messages.

Sensitivity to privacy concerns: avoids eavesdropping and stalking by malicious nodes. TruMan has not been designed with this in mind, but it does not inhibit privacy protection. However, it does require that nodes cannot be anonymous.

Robustness: the model's resistance to attacks. TruMan satisfies this property. Malicious nodes are quickly and accurately identified, making it difficult for them to perform attacks. Experiments show that, when fewer than 50% of nodes in the network are malicious, Truman performs as expected. Collusion attacks must be performed by more than half of the entire network, in which case the network is considered compromised. Furthermore, since nodes take into consideration experiences from other trustworthy nodes, a malicious node that occasionally behaves correctly can still be identified.

B. Comparison with other trust models

For the Malicious Node Identification Algorithm (MaNI) proposed in [16], the authors present a malicious node identification scheme based on strongly connected components and graph coloring. The model is proposed for complex networks in general, but it is designed only for static networks and the algorithm relies on a global observer which has information about the complete network. It is, however, very efficient thanks to the classification of nodes into components and the usage of a fast heuristic. The usage of strongly connected components and coloring serves as a basis for TruMan, which is expanded to work on distributed and dynamic networks such as vehicular networks.

The model proposed in [22] uses several criteria to judge whether or not a received message is trustworthy. First, nodes are classified by their roles, used for vehicles which should be automatically trustworthy (i.e. police cars). Nodes also store their experience each time an event message is received (if one neighboring node reported an event which did not turn out to be true, its trust value is reduced). Additionally, messages have higher reliability when their senders are closer in time and space to the reported event. When several messages about the same event are received, a node can either choose the n most trustworthy senders, according to the priority (fewer chosen nodes mean a faster, but less precise, decision), or compute the majority opinion of the messages according to each sender's trust value. However, the model relies only on direct interaction between pairs of nodes, so no form of indirect trust is considered.

In [23], the authors propose to evaluate messages utilizing a cluster-based trust model. By separating nodes into clusters with their geographical neighbors, it is possible to distribute the evaluation of messages using previously formed opinions. When a node sends a message, the cluster-leader must aggregate the other nodes' opinions on that message. Messages are only forwarded to other clusters if the aggregate opinion is above a certain threshold. Additionally, nodes that receive the message only act if the overall trust on it is above another threshold. However, it is unclear how the model behaves when the network is too sparse to form relevant clusters, neither do the authors inform how the aforementioned thresholds are decided. Furthermore, maintaining clusters in a highly dynamic network is a costly job and, if the cluster leader itself is malicious, all cluster information become untrustworthy.

The ART model proposed in [24] works in two main steps: data gathering and malicious node detection. It uses

TABLE II: Properties of Truman and related work

Property	Truman	[16]	[22]	[23]	[24]	[25]
Decentralized	✓	-	✓	✓	✓	-
Sparsity	✓	-	✓	✓	-	✓
Dynamics	✓	-	✓	✓	-	-
Scalability	✓	✓	✓	✓	-	✓
Confidence	✓	✓	✓	✓	✓	-
Security	-	-	✓	✓	-	✓
Privacy	-	-	✓	-	-	✓
Robustness	✓	-	-	-	✓	-
Efficiency	✓	✓	-	-	-	-
Cost	$\|V\| * \|E\|$	$\|V\| + \|E\|$	n/a	n/a	n/a	n/a

the Dempster-Shafter theory of evidence to merge data coming from other nodes. Then, it uses a Cosine-based metric to compare two nodes' trust vectors (a series of opinions regarding other nodes). However, these steps require several intensive calculations, which greatly increase the complexity of the algorithm. The authors present no details on how it deals with sparsity, dynamics, scalability, security and privacy.

The authors of [25] propose a cloud-based solution for a trust model, which requires an Internet-based global trust manager. This has the advantage of simplifying properties such as handling sparsity and scalability, but also makes the system slower in general, especially in situations in which mobile communication is slow or unreliable. It also makes the system prone to attacks, since the whole system collapses if the global trust manager is attacked.

Finally, it is worth noting that, aside from [16], none of the related work presents complexity calculations for its algorithm. Considering the scale of the problem, TruMan's cost of $O(|V| \times |E|)$ is very low without sacrificing completeness and correctness. The model satisfies the desired properties of a trust model, making it viable for real-world use.

V. Conclusion

The concept of trust as applied in VANETs is a powerful tool for those seeking to reduce the spread of false information as much as possible. In this paper, a new trust model for vehicular networks was presented, which combines the efficiency of previous algorithms in order to generate fast and accurate results. Nearly all malicious nodes are detected when they constitute up to 50%, with very few false positives polluting the results, without incurring substantial cost.

As nodes travel across the network and collect more data from neighbors, they are able to form an abstraction of the network which can be used to detect malicious nodes. By placing nodes into strongly connected components, a network containing a large amount of node can be simplified into a much smaller one. Using a simple graph coloring algorithm, most malicious nodes stand out by having different colors than the majority of nodes. This allows for a low complexity approach to malicious node identification in a dynamic network.

The experiments show that vehicles within a network can form a sufficient model of the network in around one day, and by then they are also able to detect nearly every malicious node in the network, with a very tiny amount of false positives. As the network changes in shape, nodes acquire more information and are able to make even more accurate classifications of malicious nodes around them. Future work includes the extension of the proposed model to use V2I communications.

References

[1] R. Viereckl, D. Ahlemann, A. Koster, E. Hirsh, F. Kuhnert, J. Mohs, M. Fischer, W. Gerling, K. Gnanasekaran, and J. Kusb, "Connected car report 2016: Opportunities, risk, and turmoil on the road to autonomous vehicles," http://www.strategyand.pwc.com/reports/connected-car-2016-study, 2016, [Online; accessed April 11, 2017].

[2] World Health Organization, "Number of road traffic deaths," 2013, [Online; accessed March 27, 2017].

[3] C. T. Barba, M. A. Mateos, P. R. Soto, A. M. Mezher, and M. A. Igartua, "Smart city for vanets using warning messages, traffic statistics and intelligent traffic lights," in *Intelligent Vehicles Symposium (IV), 2012 IEEE*. IEEE, 2012, pp. 902–907.

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

[5] D. Jiang and L. Delgrossi, "Ieee 802.11 p: Towards an international standard for wireless access in vehicular environments," in *Vehicular Technology Conference, 2008. VTC Spring 2008.*, 2008, pp. 2036–2040.

[6] J. T. Isaac, S. Zeadally, and J. S. Camara, "Security attacks and solutions for vehicular ad hoc networks," *IET communications*, vol. 4, no. 7, pp. 894–903, 2010.

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

[8] M. T. Garip, M. E. Gursoy, P. Reiher, and M. Gerla, "Congestion attacks to autonomous cars using vehicular botnets," in *NDSS Workshop on Security of Emerging Networking Technologies (SENT)*, 2015.

[9] F. D. Cunha, A. C. Vianna, R. A. Mini, and A. A. Loureiro, "How effective is to look at a vehicular network under a social perception?" in *Wireless and Mobile Computing, Networking and Communications (WiMob), 2013 IEEE 9th Inter. Conf. on*, 2013, pp. 154–159.

[10] ——, "Is it possible to find social properties in vehicular networks?" in *2014 IEEE Symposium on Computers and Communications (ISCC)*. IEEE, 2014, pp. 1–6.

[11] R. Tarjan, "Depth-first search and linear graph algorithms," *SIAM journal on computing*, vol. 1, no. 2, pp. 146–160, 1972.

[12] A. Mittal, P. Jain, S. Mathur, and P. Bhatt, "Graph coloring with minimum colors: An easy approach," in *Communication Systems and Network Technologies (CSNT), 2011 International Conference on*. IEEE, 2011, pp. 638–641.

[13] K. Appel, W. Haken, and J. Koch, "Every planar map is four colorable," *Bull. Amer. Math. Soc*, vol. 82, no. 5, pp. 711–712, 1976.

[14] A. Sánchez-Arroyo, "Determining the total colouring number is np-hard," *Discrete Mathematics*, vol. 78, no. 3, pp. 315–319, 1989.

[15] D. Brélaz, "New methods to color the vertices of a graph," *Communications of the ACM*, vol. 22, no. 4, pp. 251–256, 1979.

[16] G. Vernize, A. L. P. Guedes, and L. C. P. Albini, "Malicious nodes identification for complex network based on local views," *The Computer Journal*, vol. 58, no. 10, pp. 2476–2491, 2015.

[17] W. Li, F. Bassi, D. Dardari, M. Kieffer, and G. Pasolini, "Defective sensor identification for wsns involving generic local outlier detection tests," *IEEE transactions on Signal and Information Processing over Networks*, vol. 2, no. 1, pp. 29–48, 2016.

[18] C. A. Kerrache, A. Lakas, and N. Lagraa, "Detection of intelligent malicious and selfish nodes in vanet using threshold adaptive control," in *Electronic Devices, Systems and Applications (ICEDSA), 2016 5th International Conference on*. IEEE, 2016, pp. 1–4.

[19] A. Keränen, J. Ott, and T. Kärkkäinen, "The one simulator for dtn protocol evaluation," in *Proceedings of the 2nd international conference on simulation tools and techniques*. ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering), 2009, p. 55.

[20] F. Ekman, A. Keränen, J. Karvo, and J. Ott, "Working day movement model," in *Proceedings of the 1st ACM SIGMOBILE workshop on Mobility models*. ACM, 2008, pp. 33–40.

[21] J. Zhang, "A survey on trust management for vanets," in *2011 IEEE International Conference on Advanced Information Networking and Applications*. IEEE, 2011, pp. 105–112.

[22] U. F. Minhas, J. Zhang, T. Tran, and R. Cohen, "Towards expanded trust management for agents in vehicular ad-hoc networks," *International Journal of Computational Intelligence: Theory and Practice (IJCITP)*, vol. 5, no. 1, pp. 03–15, 2010.

[23] C. Chen, J. Zhang, R. Cohen, and P.-H. Ho, "A trust modeling framework for message propagation and evaluation in vanets," in *Information Technology Convergence and Services (ITCS), 2010 2nd International Conference on*. IEEE, 2010, pp. 1–8.

[24] W. Li and H. Song, "Art: An attack-resistant trust management scheme for securing vehicular ad hoc networks," *IEEE Transactions on Intelligent Transportation Systems*, vol. 17, no. 4, pp. 960–969, 2016.

[25] X. Chen and L. Wang, "A cloud-based trust management framework for vehicular social networks," *IEEE Access*, vol. 5, pp. 2967–2980, 2017.