Trust management for vehicular networks

Renan Greca
Departamento de Informática
Universidade Federal do Paraná
Curitiba, Brazil
Email: rdmgreca@inf.ufpr.br

Luiz Carlos Albini
Departamento de Informática
Universidade Federal do Paraná
Curitiba, Brazil
Email: 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 doesn't 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 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 will have lower trust values than the ones which appear to be reliable. This work proposes 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 and efficient, thanks to the low complexity of the algorithms 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 1 million people lose their lives to traffic accidents every year [2], so solutions to improve road safety are crucial for modern life. Vehicular ad-hoc networks are a much-studied usage of vehicular networking features. In these networks, all nodes are related to traffic; they can be vehicles equipped with on-board computers, or stationary units placed near roads. By quickly sharing data with neighboring vehicles, without the need of an Internet connection, smart vehicles can alert their drivers of important road conditions [3], while autonomous vehicles can synchronize their movements to maximize traffic throughput [4].

The accepted 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). Communications 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 VANETs and their nodes refer exclusively to vehicles and their on-board units.

As is expected for new technologies, vehicular communications can become an appealing target for malicious users and attackers. Some issues that could be exploited in a VANET 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 and stalking [6].

Each of these problems require specific solutions, although there are ways of making the network safer in general. One way is taking advantage of the concept of trust between members of the network. By having nodes remember 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 trust, which emphasizes the contents of a message, and entity-based trust, which emphasizes the sender of a message. The solution described in this article is one of entity-based trust.

The remainder of this paper is organized as follows. section II shows some examples of existing trust models for vehicular networks and the trust model for static networks that served as basis for this one; section III explains the algorithms used to develop the trust model; section IV shows the simulations which validate the usefulness of the trust model; and section V presents closing thoughts on the project.

II. Previous work

Several models have been proposed to solve the problem of trust in vehicular networks. In this section, some of the most relevant ones are described, considering the time in which they were proposed, the advantages they bring and their contributions to later study. None of them provide a complete solution, but serve as pieces of a puzzle that is still incomplete. Many trust management solutions for VANETs have been proposed over the years, such as [9], [10], [11], [12], [13], [14], [15], [16]. There are also some review and/or survey articles on the subject of VANET trust models, such as [17], [18], [19], [20], [21] [22], and [23].

For the Malicious Node Identification Algorithm (MaNI) proposed in [24], 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 is not suited for VANETs because it is designed only for static networks. Furthermore, the algorithm is executed by 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 components and coloring serve as basis for the trust model proposed here, which is expanded to work on distributed and dynamic networks such as vehicular networks.

III. Algorithm

The objective of the 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 a preset interval. In every iteration, a node checks its neighbors to see if there were changes to the network and runs a combination of algorithms that help it detect malicious nodes in the known network. Then, it separates graph T into components using Tarjan's strongly connected components algorithm. Finally, 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 trust model.

A. Tarjan's strongly connected components algorithm

An important aspect of the trust model is the use of Tarjan's strongly connected components algorithm [25]. This allows a large graph to be abstracted into a smaller graph, which therefore reduces the input for further algorithms. Given a directed graph $T = (V, O)$, a strongly connected component is defined as a group of nodes in which, for any pair of nodes $u, v \in V$, there exists 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 1. Every node of the input graph T must belong to a component.

In the implementation used, $index$, $lowlink$, $count$ and $stack$ are global variables accessible from every call of the function. $index$ and $lowlink$ are arrays indexed by node IDs (predefined unique identifiers), $count$ is an integer and $stack$ is a last-in-first-out data structure.

The algorithm works by performing a depth-first search, adding nodes to $stack$ as they are visited. If two nodes are present on $stack$, then there is a path from the first node to the second one (in the order they were added to $stack$). Each node has two attributes assigned to it during the execution of the algorithm: $index$ is used to number the nodes in the order they are visited, while $lowlink$ is the lowest indexed node reachable from each node.

In the call that visits a node u, the algorithm must loop through each node v trusted by u (that is, $u \to v$ exists and has value 1). If node v has not yet been visited, the algorithm is called for v. The $lowlink$ of u is then calculated as the smallest value between $lowlink[u]$ and $lowlink[v]$, because any node reachable from v is also reachable from u. After the loop, if $lowlink[u]$ is equal to $index[u]$, it means that u is the lowest indexed node reachable from itself and that it is the root of a component. Therefore, nodes must be popped from the $stack$ until u is found. Each node popped, including u, is a member of a strongly connected component.

The number of components is, at most, $|v|$. In a worst-case scenario, each node is put into its own component; however, this would not be the case for most useful input graphs.

Algorithm 1 shows the general structure of Tarjan's algorithm [25]. The complexity of the algorithm is $O(|V| + |O|)$ for a graph $T = (V, O)$.

With the results of Tarjan's algorithm, an undirected component graph $C = (V', O')$ is formed. Each $v' \in V'$ is the abstraction of one component identified by Tarjan's algorithm, while the edges $o' \in O'$ are edges from T between nodes that do not belong in the same component.

Algorithm 1 Tarjan's strongly connected components algorithm

function Tarjan(vertex u)
    $index[u] = count$
    $lowlink[u] = count$
    $count \leftarrow count + 1$
    push u **to** $stack$
    for v **in** neighbors of u **do**
        if weight of $u \to v$ is 0 **then**
            continue
        if $index[v] = -1$ **then** *// v has not been visited yet*
            Tarjan(v)
        $lowlink[u] \leftarrow$ **min**$(lowlink[u], lowlink[v])$
    if $lowlink[u] = index[u]$ **then**
        repeat *// unstack nodes until u is found*
            pop w **from** $stack$
            add w **to** $component$
        until $w = u$

B. Graph coloring with minimum colors

The algorithm proposed in [26] is an efficient approach to graph coloring, a classic graph theory problem. 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 presented the best results, so it has been chosen as the heuristic for the trust model.

The process of graph coloring consists of labeling each node with a color so that no two neighboring nodes share the same color. This problem has been studied in Computer Science since, at least, 1972 [27] and has been studied as a classic mathematics problem for even longer [28]. It has been proven mathematically that any planar graph can be colored with at most four colors [29], but discovering the smallest number of colors necessary to color an arbitrary graph (called the graph's chromatic number) is an NP-hard problem [30].

In [26], 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 an undirected graph $G = (V, E)$. As a comparison, the classic DSATUR algorithm for graph coloring has complexity $O(|V|^2)$ [31]. For the purposes of this study, it is not necessary to prove that the coloring algorithm's output uses the minimum possible number of colors.

Algorithm 2 shows the general structure of the graph coloring algorithm [26] [32].

A limitation of this algorithm is that the edges must be sorted according to node indexes beforehand. It does not matter which nodes get assigned which indexes, but once they are assigned those numbers, the algorithm must follow the edges in numerical order. This is demonstrated in [32].

Algorithm 2 Graph coloring with minimum colors

```
function COLORING(graph G)
    color all nodes of G with 0
    d → 0
    for e = (u, v) in edges of G do
        if u and v have the same color then
            if color[v] = d then
                d → d + 1
            color[v] → d
```

C. Trust management for a vehicular network

In order to work with dynamic networks, an algorithm must consider snapshots as inputs. The algorithm runs continuously, running new iterations at a predetermined interval. Each iteration i is associated with a timestamp, which indicates when the snapshot was taken. A snapshot $G_i = (V_i, E_i)$ represents the complete topology of the network at the iteration i; V_i is the set of nodes participating in the network at that time and E_i is the set of edges connecting pairs of nodes within communication range of each other. As the network is dynamic, it is expected that each G_i is different from G_{i-1}, but also that the changes follow a determined mobility model.

Furthermore, it the algorithm runs in a decentralized fashion, meaning each node in the network runs its own instance of the algorithm. This is necessary because it is not reasonable to expect a supervisor of the network to have complete knowledge of the nodes and relationships in the network. Each node starts knowing only about itself and maintains its own abstraction of the network surrounding it. Every node u has a static, connected and directed trust graph $T^u = (V^u, O^u)$, in which V^u is the set of nodes u is aware of and O^u is the set of trust relationships (opinions) u knows about between members of V^u. Since T^u changes over time, there is a T_i^u for every iteration i.

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

1) Node u checks who are its neighbors (nodes within communication range). Newly discovered nodes and newly formed edges are added to T_i^u. Edges are created with weight 0.5.
2) Node u tests all of 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, adding nodes and edges that were present on T_{i-1}^v but not on T_{i-1}^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 identified as benign or malicious according to the same rules as the MaNI algorithm.

During steps 1, 2 and 3, the node is collecting and organizing information. The storage of an abstraction of the network in the form of a trust graph T is crucial for the remainder of the algorithm. A prerequisite of step 2 is a test that correctly classifies a neighboring node as benign or malicious. After these steps, T_i^u is formed, which is then used for the following steps.

Each edge $w \rightarrow v$ in T_i^u is weighted according to the degree of trust w has for v with a value between 0 and 1. The closer the value is to 1, the more w trusts v. These values are not mutual, so the value of $w \rightarrow v$ can be different from the value of $v \rightarrow w$.

After the collection of data, T_i^u is separated into strongly connected components using Tarjan's algorithm [25], which is described in detail on subsection III-A. For each node in a component, there is a path formed by edges of weight higher than a threshold $0 < t < 1$ 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 $C_i^u = (V_i'^u, O_i'^u)$.

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, one or more heuristics can be used to classify the nodes as benign or malicious.

The authors of [24] describe the coloring heuristic, which can use a graph coloring algorithm such as DSATUR [31] or the algorithm described in subsection III-B [26]. After running either algorithms 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 in 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 their components' classifications.

In the experiments shown in [32], the coloring heuristic shows the most promising results, identifying a high ratio of the malicious nodes in the network. Other heuristics were experimented with, but were either less effective in detecting malicious nodes, or provided too many false positives. Therefore, for the purposes of this research, only the coloring heuristic is considered.

IV. Results

In order to test the trust model, simulations were made using an implementation of the algorithm in Python. To generate the input graphs with node mobility, the ONE simulator [33] was used in conjunction with the Working Day Movement Model [34], which provides a reasonable imitation of 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.

Most of the parameters for the simulator were taken from the article detailing the Working Day Movement Model. The simulation ran for 86400 seconds (24 hours), with a work day length of 28800 seconds and a standard deviation of departure time of 7200 seconds. Nodes move between 7 and 10 m/s in an area of approximately 14 km^2 based on a section of the map of Helsinki. There is a total of 160 nodes, 150 of which are following the Working Day Movement Model, and the other 10 are moving randomly to simulate vehicles that do not follow daily patterns. 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.

A. Network Density

The communication range of nodes vary from 10m to 50m, to illustrate the impact of different network densities. The network density is a value that abstracts the volume and frequency of connections in a vehicular network. It is calculated using the communication range of the nodes, the amount of nodes, and the total area of the simulation. For this trust model, higher densities yield better results, since nodes can construct and update their models of the network more quickly (this is demonstrated in subsection IV-B).

The simulations shown here have densities that vary between 0.001 (10m range) and 0.04 (50m range). As a comparison, the density of São Paulo was calculated as 2.24 with 10m range, a value much higher than what is necessary for the algorithm.

B. Simulations

To improve readability, all figures in this section follow the same format. The X axis shows the results of sequential iterations, ranging from 0 to 8639, while the Y axis shows a percentage of all nodes in the network, ranging from 0 to 100. The blue line represents the percentage of nodes detected

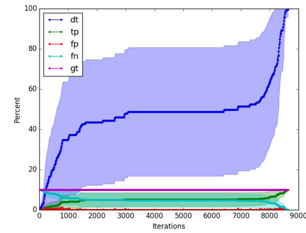


Fig. 1. Simulation with random nodes and range of 10m.

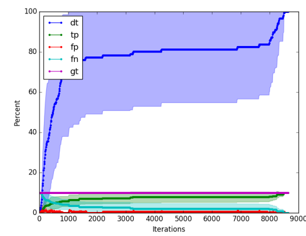


Fig. 2. Simulation with random nodes and range of 30m.

out of the complete network. Magenta is the percentage of malicious nodes in the network (ground truth). Finally, green represents the nodes correctly identified as malicious (true positives), cyan represents the undetected malicious nodes (false negatives) and red represents the benign nodes incorrectly identified as malicious (false positives).

Figure 1, Figure 2 and Figure 3 show 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 communication range allows the algorithm to converge much sooner, taking over 8000 iterations with 10m range and achieving solid results at just over 1000 iterations with 50m range.

Figure 5 to Figure 10 show 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 is expected, the results are inconsistent as control of the network is completely divided between benign and malicious nodes; at this point, the network is completely compromised. The amount of malicious nodes also affects network discovery, since nodes do not trust information from malicious neighbors.

Figure 11 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.

V. Conclusion

While malicious nodes can diminish the usefulness of vehicular networks, the concept of trust as applied in VANETs is a powerful tool for those seeking to reduce the spread of

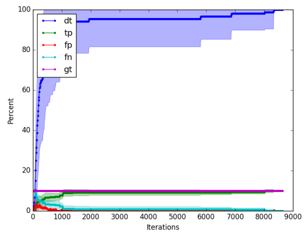


Fig. 3. Simulation with random nodes and range of 50m.

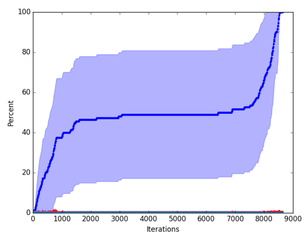
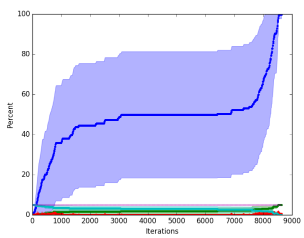


Fig. 4. Simulation with random nodes, range of 10m and 1% malicious.



0.5

Fig. 5. Simulation with random nodes, range of 10m and 5% malicious.

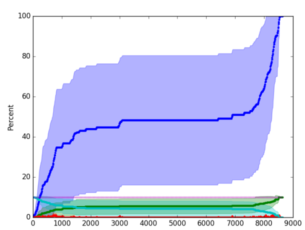


Fig. 6. Simulation with random nodes, range of 10m and 10% malicious.

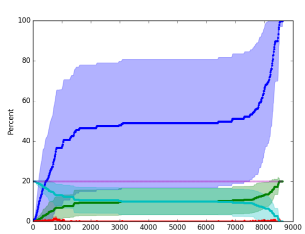


Fig. 7. Simulation with random nodes, range of 10m and 20% malicious.

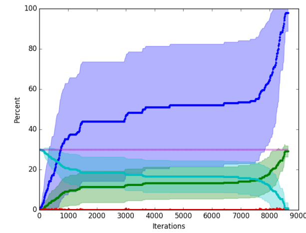


Fig. 8. Simulation with random nodes, range of 10m and 30% malicious.

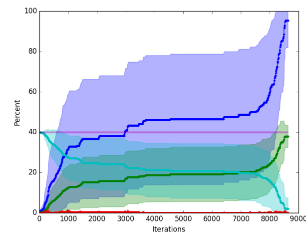


Fig. 9. Simulation with random nodes, range of 10m and 40% malicious.

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.

As nodes travel across the network and collect more data from neighbor, they are able to form an abstraction of the network which can be used to detect malicious nodes. By placing nodes into strongly connected component with Tarjan's algorithm, a network containing a large amount of node can be simplified into a much smaller one. And, using a simple graph coloring algorithm, most malicious nodes stand out by

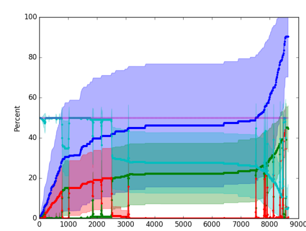


Fig. 10. Simulation with random nodes, range of 10m and 50% malicious.

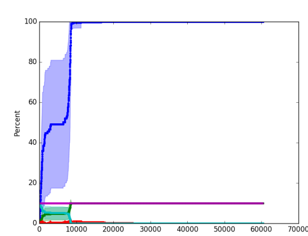


Fig. 11. Simulation with random nodes, range of 10m and 10% malicious during 7 days.

having colors different than the majority of nodes.

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 small amount of false positives. Since trust amongst nodes in the network is a value ranging between 0 and 1, the trust between two nodes gradually increases or decreases according to the results of each iteration.

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," http://www.who.int/gho/road_safety/mortality/traffic_deaths_number/en/, 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. IEEE*. IEEE, 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), San Diego, CA*, 2015.

[9] A. Patwardhan, A. Joshi, T. Finin, and Y. Yesha, "A data intensive reputation management scheme for vehicular ad hoc networks," in *Mobile and Ubiquitous Systems: Networking & Services, 2006 Third Annual International Conference on*. IEEE, 2006, pp. 1–8.

[10] M. Gerlach, "Trust for vehicular applications," in *Autonomous Decentralized Systems, 2007. ISADS'07. Eighth International Symposium on*. IEEE, 2007, pp. 295–304.

[11] M. Raya, P. Papadimitratos, V. D. Gligor, and J.-P. Hubaux, "On data-centric trust establishment in ephemeral ad hoc networks," in *INFOCOM 2008. The 27th Conference on Computer Communications. IEEE*. IEEE, 2008, pp. 1238–1246.

[12] D. Huang, X. Hong, and M. Gerla, "Situation-aware trust architecture for vehicular networks," *IEEE Communications Magazine*, vol. 48, no. 11, 2010.

[13] Q. Ding, X. Li, M. Jiang, and X. Zhou, "A novel reputation management framework for vehicular ad hoc networks," *International Journal of Multimedia Technology*, vol. 3, no. 2, pp. 62–66, 2013.

[14] N. Haddadou, A. Rachedi, and Y. Ghamri-Doudane, "Trust and exclusion in vehicular ad hoc networks: an economic incentive model based approach," in *Computing, Communications and IT Applications Conference (ComComAp), 2013*. IEEE, 2013, pp. 13–18.

[15] Z. Liu, J. Ma, Z. Jiang, H. Zhu, and Y. Miao, "Lsot: A lightweight self-organized trust model in vanets," *Mobile Information Systems*, vol. 2016, 2016.

[16] 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.

[17] 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.

[18] S. Ma, O. Wolfson, and J. Lin, "A survey on trust management for intelligent transportation system," in *Proceedings of the 4th ACM SIGSPATIAL International Workshop on Computational Transportation Science*. ACM, 2011, pp. 18–23.

[19] J. Zhang, "Trust management for vanets: challenges, desired properties and future directions," *International Journal of Distributed Systems and Technologies (IJDST)*, vol. 3, no. 1, pp. 48–62, 2012.

[20] M. N. Mejri, J. Ben-Othman, and M. Hamdi, "Survey on vanet security challenges and possible cryptographic solutions," *Vehicular Communications*, vol. 1, no. 2, pp. 53–66, 2014.

[21] S. A. Soleymani, A. H. Abdullah, W. H. Hassan, M. H. Anisi, S. Goudarzi, M. A. R. Baee, and S. Mandala, "Trust management in vehicular ad hoc network: a systematic review," *EURASIP Journal on Wireless Communications and Networking*, vol. 2015, no. 1, p. 146, 2015.

[22] J. S. Sengar, "Survey: Reputation and trust management in vanets," *International Journal of Grid and Distributed Computing*, vol. 9, no. 1, pp. 201–206, 2016.

[23] S. Dwivedi and R. Dubey, "Review in trust and vehicle scenario in vanet," *International Journal of Future Generation Communication and Networking*, vol. 9, no. 5, pp. 305–314, 2016.

[24] 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.

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

[26] 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.

[27] R. M. Karp, "Reducibility among combinatorial problems," in *Complexity of computer computations*. Springer, 1972, pp. 85–103.

[28] A. B. Kempe, "On the geographical problem of the four colours," *American journal of mathematics*, vol. 2, no. 3, pp. 193–200, 1879.

[29] 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.

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

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

[32] G. Vernize, "Identificação de nós maliciosos em redes complexas baseada em visões locais," MSc dissertation, Universidade Federal do Paraná, 2013.

[33] 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.

[34] 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.