Expert Systems With Applications 50 (2016) 40–54

Contents lists available at ScienceDirect

Expert Systems With Applications

journal homepage: www.elsevier.com/locate/eswa

CEAP: SVM-based intelligent detection model for clustered vehicular ad hoc networks

Omar Abdel Wahab[a], Azzam Mourad[a,*], Hadi Otrok[b,c], Jamal Bentahar[c]

[a] Department of Computer science and Mathematics, Lebanese American University, Beirut, Lebanon
[b] Department of Electrical & Computer Engineering, Khalifa University of Science, Technology & Research, Abu Dhabi, UAE
[c] Concordia Institute for Information Systems Engineering, Montreal, Canada

ARTICLE INFO

Keywords:
Vehicular ad hoc network
Intrusion detection
High mobility
Support vector machine (SVM)
Malicious node
Training set size reduction

ABSTRACT

The infrastructureless and decentralized nature of Vehicular Ad Hoc Network (VANET) makes it quite vulnerable to different types of malicious attacks. Detecting such attacks has attracted several contributions in the past few years. Nonetheless, the applicability of the current detection mechanisms in the deployed vehicular networks is hindered by two main challenges imposed by the special characteristics of VANETs. The first challenge is related to the highly mobile nature of vehicles that complicates the processes of monitoring, buffering, and analyzing observations on these vehicles as they are continuously moving and changing their locations. The second challenge is concerned with the limited resources of the vehicles especially in terms of storage space that restricts the vehicles' capacity of storing a huge amount of observations and applying complex detection mechanisms. To tackle these challenges, we propose a multi-decision intelligent detection model called CEAP that complies with the highly mobile nature of VANET with increased detection rate and minimal overhead. The basic idea is to launch cooperative monitoring between vehicles to build a training dataset that is analyzed by the Support Vector Machine (SVM) learning technique in online and incremental fashions to classify the smart vehicles either cooperative or malicious. To adapt the proposed model to the high mobility, we design it on top of the VANET QoS-OLSR protocol, which is a clustering protocol that maintains the stability of the clusters and prolongs the network's lifetime by considering the mobility metrics of vehicles during clusters formation. To reduce the overhead of the proposed detection model and make it feasible for the resource-constrained nodes, we reduce the size of the training dataset by (1) restricting the data collection, storage, and analysis to concern only a set of specialized nodes (i.e., Multi-Point Relays) that are responsible for forwarding packets on behalf of their clusters; and (2) migrating only few tuples (i.e., support vectors) from one detection iteration to another. We propose as well a propagation algorithm that disseminates only the final decisions (instead of the whole dataset) among clusters with the aim of reducing the overhead of either exchanging results between each set of vehicles or repeating the detection steps for the already detected malicious vehicles. Simulation results show that our model is able to increase the accuracy of detections, enhance the attack detection rate, decrease the false positive rate, and improve the packet delivery ratio in the presence of high mobility compared to the classical SVM-based, Dempster–Shafer-based, and averaging-based detection techniques.

1. Introduction

Road accidents constitute the main leading cause of death for young people aged between 15 and 29 and the eight leading cause of mortality in general according to the World Health Organization (WHO)[1]. The organization warned that in 2030, road death will probably become the fifth-leading reason of death if precautionary measures are not taken. Vehicular Ad Hoc Network (VANET) (Wahab, Otrok, & Mourad, 2013b) is a multi-agent wireless network that is designed mainly to solve the traffic problems by allowing the smart vehicles to communicate with each other, as well

* Corresponding author. Tel.: +96170343643, +961 1 786456x1200; fax: +961 1 867098.
E-mail addresses: omar.abdelwahab@lau.edu.lb (O.A. Wahab), azzam.mourad@lau.edu.lb, azzammourad@gmail.com (A. Mourad), Hadi.Otrok@kustar.ac.ae (H. Otrok), bentahar@ciise.concordia.ca (J. Bentahar).

http://dx.doi.org/10.1016/j.eswa.2015.12.006
0957-4174/

[1] Global status report on road safety 2013: Supporting a decade of action, published by World Health Organization.

as with the roadside infrastructure such as traffic lights. For example, VANET enables vehicles to exchange emergency alerts in order to avoid collisions. Nevertheless, the applications of VANET are not restricted to the safety applications but englobe as well marketing, multi-media, and infotainment services (Huang, Chen, Chen, & Wu, 2009). Therefore, providing a good level of Quality of Service (QoS) is essential in such networks in order to ensure timely and accurate message delivery. Moreover, the highly mobile topology of vehicles imposes to take into consideration the mobility metrics in VANET to maintain the stability of the network.

Vehicular Ad Hoc Network Quality of Service Optimized Link State Routing (VANET QoS-OLSR) (Wahab et al., 2013b) is a clustering protocol that considers a tradeoff between the QoS requirements and the high mobility metrics in VANET. The protocol is based on electing a set of optimal cluster-heads in terms of QoS and dividing the network into clusters. To this end, a QoS function composed of several combinations of both QoS-based (bandwidth, connectivity) and mobility-based (velocity, residual distance) metrics is defined. The idea is to form stable clusters without sacrificing the QoS requirements. This function is used to elect the cluster-heads whose advantage is to facilitate the management of the clusters (Cheng, Yang, & Cao, 2013). These heads are then responsible for selecting a set of specific vehicles charged of transmitting the network topology information through messages called *Topology Control (TC)* and forwarding the packets. Such nodes are called *MultiPoint Relay (MPR)* nodes. VANET QoS-OLSR uses an algorithm based on Ant Colony Optimization (ACO) (Dorigo, Caro, & Gambardella, 1999) to select the MPRs satisfying the optimal path constraints. This algorithm takes into consideration the QoS function and End-to-End delay for this purpose. However, the problem arises when these selected intelligent MPRs behave maliciously and begin launching several attacks for the purpose of disrupting the network. Therefore, we propose in this paper an intelligent detection mechanism based on Support Vector Machine (SVM) learning technique to classify the vehicles in the clustered Vehicular Ad Hoc Networks either cooperative or malicious, while considering VANET QoS-OLSR as a starting point. The reason behind considering VANET QoS-OLSR comes from the fact that this protocol considers the formation of stable and long-living clusters, which is necessary in VANETs for any monitoring mechanism that requires buffering and comparing messages. As a case study, the packet dropping attack in which malicious MPRs drop the packets supposed to be retransmitted is considered. Such a misbehavior degrades the performance and lifetime of the network by isolating some cluster-heads. These cluster-heads will no longer receive the *TC* messages and hence will not be able to communicate with the other heads, which leads to a disconnected network. Although the packet dropping attack is considered as a case study in the subsequent sections, our model is generic and can be adapted to detect different types of malicious behaviors (e.g., Identity spoofing, Wormhole, etc.) by modifying the attributes used to build the classifiers accordingly.

The existing approaches that tackle the problem of malicious nodes in the domain of networks can be divided into two parts: detection-oriented approaches whose main goal is to identify the malicious nodes, and reaction-oriented approaches whose main goal is to deal with nodes after detection. This paper addresses the challenging problem of detecting and identifying the malicious vehicles in VANET, which is still an open research problem because of the challenges that are imposed by the special characteristics of VANET on any proposed detection mechanism. As for the reaction part, we have already proposed in our previous work (Wahab, Otrok, & Mourad, 2013a) a modified Tit-for-Tat strategy that could be adopted on top of our proposed detection model to control the relationships between vehicles after detection. The proposed strategy regulates the cooperation between vehicles in VANET after detection by propagating the detection results and advising the network nodes to fulfill the requests incoming from the detected cooperative vehicles and to drop those incoming from detected misbehaving vehicles.

Numerous detection models have been proposed in the literature for detecting misbehaving nodes in VANET. Nonetheless, the applicability of the existing models is hindered by the challenging characteristics of VANET. Specifically, the high mobility of vehicles complicates the process of monitoring, buffering, and analyzing observations as vehicles are continuously moving and changing their locations. Moreover, the limited resources of the vehicles especially in terms of storage space restricts the ability of vehicles to store and analyze the huge amount of observations that may be needed for accurate detections. To tackle these problems, we propose in this paper a cluster-based intelligent detection model for malicious vehicles called CEAP (Collection, Exchange, Analysis, and Propagation). The model combines the SVM classification technique (Han, Kamber, Pei, & Kaufmann, 2012) and watchdogs monitoring concept (Marti, Giuli, Lai, & Baker, 2000) in order to optimize the decision making process. The reasons behind choosing SVM as a classification technique are that (1) it is commonly known to be the best machine learning technique for binary (two classes) classification (Heller, Svore, Keromytis, & Stolfo, 2003; Hu, Liao, & Vemuri, 2003; Konar, Chakraborty, & Wang, 2005; Massimiliano, Alessandro, & Roi, 1997; Sung & Mukkamala, 2003; Vapnik, 1995); (2) it has been successfully used for intrusion detection (Heller et al., 2003; Hu et al., 2003; Sung & Mukkamala, 2003); (3) it is effective in high dimensional datasets with a large number of attributes (Shon & Moon, 2007); (4) unlike some other machine learning techniques such as Neural Networks (Haykin, 1998), SVM yields a unique solution since the optimality problem in SVM is convex (Auria & Moro, 2008); (5) it produces very accurate classifiers, is robust to noise, and minimizes the overfitting (Han et al., 2012).

In CEAP, the database is a result of a cooperative watchdogs monitoring process in which the watchdogs gather and share evidences about the behavior of the vehicles being classified. The SVM is then used by the cluster members in a distributed manner to distinguish well-behaving from misbehaving MPRs. For the sake of increasing the accuracy, we adapt SVM to work in both incremental and online fashion. This means that the training set is continuously growing by adding new training tuples (evidences) at each iteration and updated by what is learned from the previous iterations, which allows us to include additional training data without re-training from scratch (Laskov, Gehl, Krüger, & Müller, 2006). In order to mitigate the overhead and increase the efficiency of the model, CEAP exploits an important property of SVM, which states that only the support vectors (essential training tuples) are used to differentiate between classes. Thus, we consider that only these support vectors are kept from one iteration to another, which reduces the training set size in a considerable manner. In addition, the data is collected at the cluster-level targeting solely the MPR vehicles within each cluster, which are a set of specialized nodes responsible for packet forwarding. Thus, the database containing observations on the MPR vehicles exclusively is only communicated among the cluster members and only the final decisions are communicated among clusters.

Contributions. In summary, our contribution is a cluster-based lightweight intelligent detection model that uses the SVM machine learning technique in incremental and online fashion to classify the smart vehicles in multi-agent VANETs either cooperative or malicious. This model is able to:

- Increase the accuracy of detections, reduce the false alarms, and improve the routing process by cooperatively collecting

a representative set of evidences and analyzing them using online SVM to come up with final and unified decisions.
- Adapt well to the high mobility of vehicles by using a cluster-based architecture designed on top of the VANET QoS-OLSR clustering protocol, which is able to improve the stability of the clusters and prolong the lifetime of the vehicular network.
- Reduce the storage, computation, and communication overhead by using a cluster-based model architecture, employing SVM in an incremental manner where only the support vectors are kept throughout iterations, and proposing a data propagation algorithm to disseminate the final decisions among clusters.

The rest of the paper is organized as follows. Section 2 reviews the related work. Section 3 formulates the problem. Section 4 explains our proposed detection model in details and presents the relevant algorithms. Section 5 explains the scenario and parameters used to perform the simulations and presents empirical results. Finally, Section 6 concludes the paper.

2. Related work

Security issues have been widely investigated in the context of VANETs (Daza, Domingo-Ferrer, Sebe, & Viejo, 2009; Engoulou, Bellaïche, Pierre, & Quintero, 2014; Lin, Sun, Ho, & Shen, 2007; Liu, Yuen, Au, & Susilo, 2014; Zhang, Wu, Solanas, & Domingo-Ferrer, 2010). One of the important challenges in this regard is the existence of misbehaving vehicles. Several approaches have been proposed in the literature to tackle this issue in both mobile and vehicular ad hoc networks. These approaches are usually categorized into (1) credit-based approaches (Douceur & Moscibroda, 2007; Lee, Pan, Park, Gerla, & Lu, 2007; Lee, Park, Gerla, & Lu, 2012; Li & Wu, 2009; Zhong, Yang, & Chen, 2003) in which the nodes pay to get served and get paid versus serving the other nodes, and (2) reputation-based approaches (Buchegger & Boudec, 2002; Lian et al., 2008; Marti et al., 2000; Michiardi & Molva, 2002) which are based on monitoring the nodes and propagating the misbehaving ones in order to isolate them. Since our proposed model falls under the second category, we survey in the following the main reputation-based approaches proposed for mobile and vehicular networks. We present as well the main contributions that used the Support Vector Machine for anomaly detection in wireless networks since our model uses SVM for nodes' classification. Finally, we discuss the limitations of the existing detection models in wireless networks and highlight the unique features of our proposed model.

2.1. Reputation-based approaches

In Marti et al. (2000), Marti et al. integrated the watchdog and pathrater idea to the Dynamic Source Routing (DSR) Johnson and Maltz (1996) protocol. This approach encompasses two modules: watchdog and pathrater. The watchdog is used to monitor the next-hop in order to ensure that intermediate nodes are retransmitting the received packets. The role of pathrater is to rate each path according to the observations received from the watchdogs and choose the best path that avoids the misbehaviors. The main shortcoming of this approach is that the misbehaving nodes are rewarded instead of being punished since they will no longer forward packets but their packets will remain being forwarded. Moreover, the efficiency of watchdogs may be hindered by packet collisions, false alarms, and partial dropping scenario.

CONFIDANT (Buchegger & Boudec, 2002) is made up of four components: monitor, reputation system, path manager, and trust manager. Each node monitors its one-hop away neighbors continuously and reports the suspicious events to the reputation system, which modifies the ratings of these suspected nodes according to the gravity and the frequency of these events. The path manager intervenes when the rating of a certain node becomes intolerable by managing the route cache accordingly. Then, the trust manager has to propagate *Alarm* message in order to warn the other nodes. However, the credibility of the generated alarms is questionable in the sense that the nodes may mislead and report false events to promote/demote some other nodes according to their objectives.

In CORE Michiardi and Molva (2002), the authors aim to enforce the cooperation of the nodes by distributing the available resources in the network according to each node's reputation, which is built according to the contribution of this node. To this end, they define a weighted reputation formula composed of three types of reputations: (1) a subjective reputation calculated based on the explicit observations, (2) an indirect reputation representing the positive information reported by others, and (3) a functional reputation linked to a specific task (i.e., packet forwarding). The main problem of this approach is that it considers only positive indirect information, which allows misbehaving nodes to increase each other's reputations by reporting bogus positive information (kaushik & Singhaii, 2011).

In Tit-for-Tat Lian et al. (2008), each node monitors its neighbor's behavior and mimics this behavior to decide whether to cooperate or defect. Nonetheless, two main limitations encounter this strategy. First, the ambiguity in the monitoring caused by the packet collisions and the high mobility of nodes would lead to false detections. In addition, this strategy leads to a deadlock phase where no node cooperates with any other node since at a certain time, each node will have a bad history of all other nodes (Wahab et al., 2013a).

Gantsou (2015) proposed a detection technique for Sybil attacks without the need for pre-registration of vehicles' identities, which helps preserve nodes' privacy. The proposed technique is based on detecting which IP addresses are related and which of them are likely to belong to the same Network Interface Card.

Khan, Agrawal, and Silakari (2015) proposed a detection scheme for the malicious nodes that drop and duplicate packets. The basic idea is that vehicles are monitored by some other trusted nodes called verifiers that specify a selection threshold above which a node is considered malicious. Thereafter, verifiers report the identifiers of the malicious vehicle to a third party called Certificate Authority (CA), which is responsible for managing the identities of vehicles and verifying the misbehavior reports conveyed by verifiers.

Sedjelmaci and Senouci (2015) proposed an intrusion detection framework for VANET called AECFV and composed of three modules: Local Intrusion Detection System (LIDS), Global Intrusion Detection System (GIDS) and Global Decision System (GDS). The LIDs runs at each cluster member's level and allows them to monitor the behavior of their cluster-head and neighbors locating with the same transmission range. The GIDS runs at each cluster-head's level and allows it to monitor the behavior of its cluster members and make final decisions concerning the suspected vehicles reported by the LIDS. Finally, the GDS runs at each Roadside Unit's level and enables it to aggregate the reputations of the vehicles based on the information that they provide and compute their appropriate trust level (Wahab, Bentahar, Otrok, & Mourad, 2015).

Kumar and Chilamkurti (2014) advocated a Learning Automata (LA) based intrusion detection system, where an automaton is embedded in each vehicle. The automaton takes as inputs from the environment the density, mobility and direction of motion of the vehicles to predict their behavior. Moreover, a Collaborative Trust Index (CTI) for each action executed by the automaton is calculated; based on which this automaton receives either penalty or reward from its neighbors.

Wahab et al. (2013a) introduced a modified Tit-for-Tat strategy called Dempster-Shafer Tit-for-Tat. This strategy works in a

cooperative manner where a set of observations are cooperatively gathered by some watchdogs and then aggregated by the cluster-head using the Dempster–Shafer theory of evidence (Chen & Venkataramanan, 2005) to come up with a final decision on the behavior of the vehicles. Using the Tit-for-Tat concept, the cluster members cooperate with the vehicles classified as cooperative and refrain from cooperating with those classified as malicious.

2.2. SVM-based approaches for anomaly detection in wireless networks

In Li, Anupam, and Tim (2010), the authors proposed SVM-based Misbehavior Detection and Trust Management framework (SMART) where SVM is used to differentiate between well-behaving and misbehaving nodes in MANETs. The authors propose also a multidimensional trust management scheme (Wahab et al., 2015) to evaluate the trustworthiness of the nodes from several perspectives. However, the limitation of SMART is that the authors did not describe how the dataset is collected to build the SVM classifier, which makes their model lack realism especially in the highly mobile networks such as MANET or VANET, where collecting accurate evidences is a challenging issue.

Kaplantzis, Shilton, Mani, and Sekercioglu (2007) introduced a centralized detection approach based on SVM where the intrusion detection systems that are run in the central station use one-class SVM to train the collected training dataset. The dataset is composed of attributes related to the bandwidth and hop counts to detect black hole and selective forwarding attacks. Nonetheless, this model yields a low detection rate for the selective forwarding attack when the occurrence of this attack is small. Moreover, due to the centralized architecture of this model, the base station has to analyze a huge number of data, which leads to a high storage, computation, and communication overhead. Therefore, this model is not suitable for the resource-constrained nodes such as mobiles (limited energy, bandwidth) or vehicles (limited bandwidth, storage space).

In Flouri, lozano, and Tsakalides (2008), the authors proposed a distributed model for training SVM in the wireless sensor networks. The model is composed of two algorithms that try to achieve a tradeoff between the amount of data diffused and the energy consumption of the nodes. The first algorithm, called Minimum Selective Gossip (MSG-SVM), aims at minimizing the amount of data propagated in order to reduce the energy consumption. The second algorithm called, Sufficient Selective Gossip (SSG-SVM), aims at determining the sufficient amount of data to be propagated in order to reach the optimality in the classification. Simulation results show that the second algorithm, which requires more data and hence more energy consumption, gives higher classification accuracy. The problem of this model is that the nodes have to analyze a huge dataset coming from different nodes in the network. In contrary, our proposed model works on the cluster level in the sense that the data is only propagated among the cluster members, while only the final decisions (not the data) is communicated among clusters.

2.3. Comparison of our model with the state-of-the-art detection models in VANET

The existing detection mechanisms suffer from two main drawbacks that limit their effectiveness in the real deployed VANETs. In fact, these mechanisms overlook a paramount characteristic that distinguishes VANET from the other types of wireless networks, which is the high mobility of nodes. Practically, the vehicles in VANET are continuously and speedily moving and changing their locations. This complicates the processes of monitoring, buffering, and analyzing data related to their behavior. To tackle this issue, we propose a cluster-based detection and design it on top of the VANET QoS-OLSR clustering protocol, which is able to prolong the network's lifetime and maintain the clusters' stability by grouping the vehicles having convergent residual distance towards destination and velocity scales into homogeneous clusters. This allows vehicles to have enough time to monitor and analyze data relating to their peers in the same cluster.

The second drawback of the most of the existing detection mechanisms, especially those that are based on SVM, is the high overhead that they entail for resource-constrained nodes such as vehicles. In fact, the existing SVM-based detection mechanisms in wireless networks are either centralized placing all the load on a single resource-constrained node or distributed requiring the training dataset to be gathered, propagated, stored, and analyzed by/to a large number of nodes. To tackle this issue, we reduce the training set data size by (1) restricting the data collection, storage, and analysis to concern only a set of specialized nodes (i.e., MPRs); and (2) migrating only few essential tuples (i.e., support vectors) from one detection iteration to another. We propose as well a data propagation algorithm that disseminates only the final decisions (instead of the whole dataset) among clusters with the aim of reducing the overhead of either exchanging results between each set of vehicles or repeating the detection mechanism steps for the already detected malicious vehicles.

Furthermore, CEAP is able to improve the detection rate by using SVM in online and incremental fashions. Specifically, support vectors are migrated from one iteration to another in the sense that the training set is continuously populated with additional training tuples, which allows us to include additional training data without re-training from scratch. Moreover, SVM is used in an online fashion so that the classes of the support vectors at each iteration $t+1$ are updated by what is learned in the previous iteration t, which allows improving the classification accuracy.

In summary, our proposed detection model enjoys three main advantages over the state-of-the-art detection models in wireless networks, which are:

- Ability to effectively operate in a highly mobile environment.
- Minimization of the storage, communication, and computation overhead by reducing the size of the analyzed training dataset.
- Improvement in the accuracy and attack detection rates by using SVM in online and incremental fashions.

3. Problem statement

VANET QoS-OLSR (Wahab et al., 2013b) is a clustering protocol that considers a tradeoff between the QoS requirements and the high mobility metrics in VANET. The basic idea of VANET QoS-OLSR is to form stable clusters in VANET and maintain the stability during communications and link failures without sacrificing the Quality of Service requirements. This is done by considering, in addition to the usual QoS parameters such as connectivity and bandwidth, the mobility metrics of the vehicles such as velocity and residual distance ratios when formulating the QoS function. The velocity ratio represents the vehicle's velocity w.r.t the average speed limits on the road and the residual distance ratio measures how far is the vehicle from reaching its intended destination. This QoS value is then computed using the following formula: $QoS(i) = BW(i) \times N(i) \times \frac{DistRatio(i)}{VelRatio(i)}$, where $QoS(i)$ denotes the QoS value of vehicle i, $BW(i)$ denotes vehicle i's bandwidth share, $N(i)$ denotes vehicle i's number of direct neighbors, $DistRatio(i)$ denotes vehicle i's distance ratio, and $VelRatio(i)$ denotes vehicle i's velocity ratio. For example, the QoS value for a vehicle v having a bandwidth share of 520 bps, 4 direct neighbors, distance ratio of 0.3, and velocity ratio of 0.4 is $QoS(v) = 520 \times 4 \times \frac{0.3}{0.4} = 1560$. This QoS function is used then to elect/select the cluster-heads/MPRs

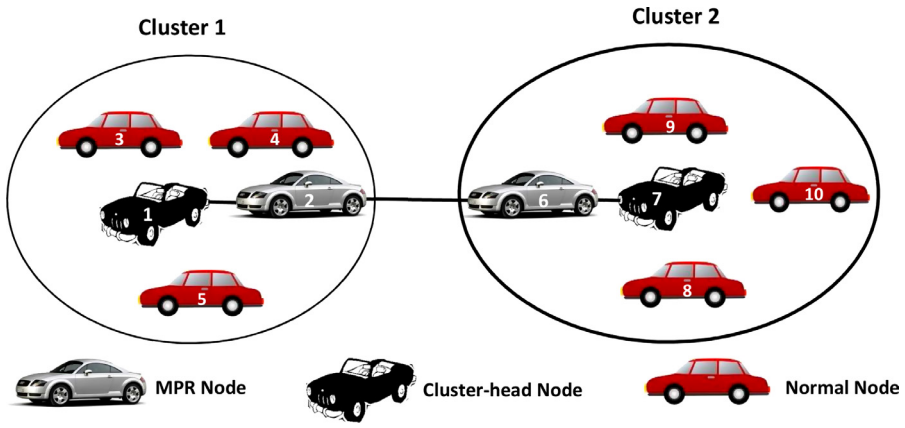


Fig. 1. VANET QoS-OLSR: vehicles 1 and 7 are elected as cluster-heads and vehicles 2 and 6 are selected as MPRs.

Table 1
QoS metrics values of vehicles.

Nodes	1	2	3	4	5
QoS value	746.5	700.56	535.2	372.6	318.7
Nodes	6	7	8	9	10
QoS value	754.8	797.8	403	236.01	159.34

Table 2
The pheromone probability values for the different paths.

Path	$p1$	$p2$	$p3$	$p4$
Nodes	5–9	5–6	2–9	2–6
QoS	554.71	1073.5	936.57	1455.36
End-to-End delay (seconds)	230	198	234	120
Pheromone	324.71	875.5	702.57	1335.36

and create the clusters. The VANET QoS-OLSR clustering algorithm, which includes cluster-heads election and MPRs selection, works as follows. First, HELLO messages (Clausen, Jacquet, Laouiti, Qayyum, & Viennot, 2002) containing the QoS values are exchanged among the neighboring vehicles. Next, each vehicle votes for the neighbor having the local maximal QoS value as cluster-head. Note that the vehicle may vote for itself if it has the highest local QoS value. According to Table 1 and Fig. 1, vehicles 1 and 7 should be elected as cluster-heads since they have the maximal QoS value among their 1-hop away neighbors. Thus, two clusters are formed: Cluster 1 composed of vehicles 1, 2, 3, 4, 5, and Cluster 2 composed of vehicles 6, 7, 8, 9, 10. In order to connect the clusters and allow them to communicate with each other, a set of vehicles called Multi-points relay (MPRs) are selected by the cluster-heads.

VANET QoS-OLSR proposes a selection procedure that is based on the Ant Colony Optimization (ACO) algorithm (Dorigo et al., 1999). The algorithm can be summarized as follows. First, the source cluster-head sends a set of ant messages responsible for route discovery. These ants visit all the possible paths towards the destination cluster-head and gather information concerning the QoS available on each path as well as the route time of these paths. Next, these cluster-heads calculate a pheromone value for each path according to the gathered information ($Pheromone(i) = QoS(i) - routeTime(i)$). Then, each cluster-head selects the vehicles belonging to the path having the highest pheromone value and locating within its cluster scope as MPRs. In our example, to connect the cluster-heads 1 and 7, there are four possible paths: p1: 5–9, p2: 5–6, p3: 2–9, and p4: 2–6. According to Table 2, the path $p4$ composed of vehicles 2 and 6 gives the higher pheromone value among the other possible paths. Therefore, the cluster-head

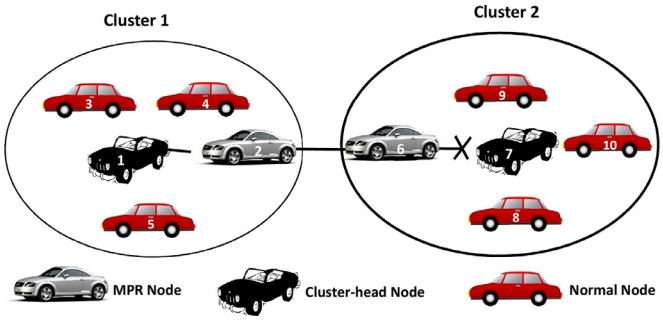


Fig. 2. Packet dropp attack example: The MPR 6 performs a packet drop attack leading to isolate the cluster-head 7.

1 selects the vehicle 2 as MPR and the cluster-head 7 selects the vehicle 6 as MPR. The problem in this protocol arises when the MPRs, after being selected, launch a packet dropping attack and decide to discard the packets instead of relaying them. The packet drop attack is a sort of Denial of Service (DoS) (Aad, Hubaux, & Knightly, 2008) where the nodes supposed to forward the packets discard them instead. The nodes can also apply this attack selectively either by dropping the packets heading to a particular destination, or by dropping a packet every p packets or every s seconds. Such misbehavior will degrade the performance and lifetime of the network by isolating some cluster-heads. These cluster-heads will no longer receive the Topology Control (TC) messages and will not be able hence to communicate with the other heads, which leads to a disconnected network. Fig. 2 shows a packet drop attack example where the MPR 6 supposed to relay packets to the cluster-head 7 discards these packets instead. This leads to isolate the cluster-head 7 and prevents it from receiving the *TC* messages and being hence aware of the network topology. Thus, cluster 2 is disconnected from the network. Fig. 3 shows how the the percentage of dropped packets (Eq. 1) increases with the increase in the percentage of malicious vehicles.

$$\textit{Percentage of dropped packets} = 100\% \times \frac{\textit{number of dropped packets}}{\textit{Total number of packets}} \quad (1)$$

The figure shows that up to 55% of the total packets may be dropped when the percentage of malicious vehicles reaches 50%, which will lead to a widely disconnected and short-living network. In order to simulate Fig. 3, the number of vehicles used is 500, the packet drop percentage varies from 1% to 100%, and the percentage of malicious vehicles varies from 10% to 50%. The percentage of dropped packets is obtained by dividing the

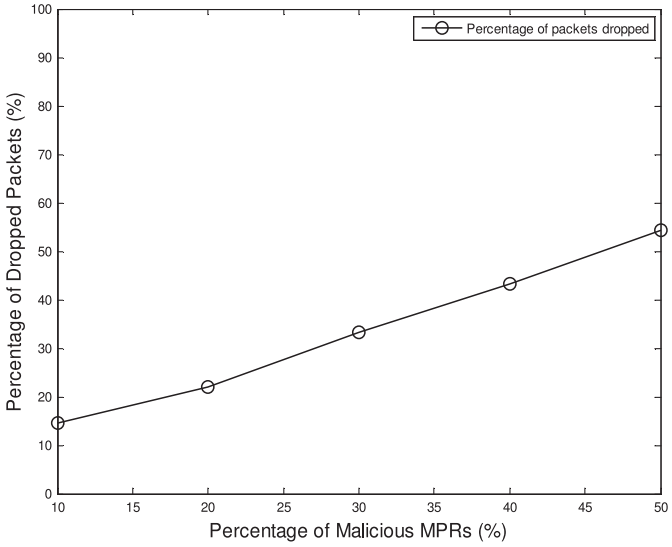


Fig. 3. The percentage of packets dropped increases with the increase in the percentage of malicious vehicles.

number of dropped packets over the total number of packets supposed to be forwarded. The number of dropped packets is calculated by incrementing this counter each time the malicious vehicle drops a number of packets according to its packet drop percentage (these simulation parameters are explained in details in Section 5). Therefore, it is highly important to develop a detection model that is able to accurately detect and identify these malicious vehicles with minimal false alarms and overhead.

4. CEAP detection model

In this section, we describe our model, called CEAP (Collection, Exchange, Analysis, and Propagation), proposed to detect the malicious vehicles in the clustered VANETs. Recall that the packet dropping attack is considered as a case study. The model is composed of four phases: (1) data collection, (2) data exchange, (3) data analysis, and (4) data propagation. The solution can be summarized as follows. In the data collection phase, the cluster members, including the cluster-head, are designated as watchdogs to continuously monitor and analyze the behavior of the MPR nodes. These watchdogs collect evidences according to their observations. The data analysis phase, which consists of analyzing the training set using SVM and classifying the MPRs accordingly, runs each t iterations and can be called for by the watchdogs to run out of the regular iterations (i.e., in the time between iteration t and iteration $t+1$) each time these watchdogs suspect a malicious behavior. In the data propagation phase, the cluster-head propagates the classes determined within its cluster to the other clusters whenever a contact between them takes place in order to mitigate the detection time and overhead. It is worth noting that the cluster-heads may also be malicious themselves by propagating falsified data. This problem is out of the scope of this paper and will be investigated as future work. The details of each phase are described in what follows.

4.1. Data collection

In this phase, the cluster-members, including the cluster-head, are designated as *watchdogs* (Marti et al., 2000) to continuously monitor and analyze the behavior of the MPR vehicles that are serving them. These watchdogs can overhear the communications that take place among the vehicles located within their transmission ranges. Thus, if a vehicle V can overhear the incoming and outgoing transmissions from/to an MPR x, then V may be assigned the role of a watchdog to monitor the behavior of x.

To this end, each watchdog maintains a buffer containing the recently sent packets from the sources (willing their packets to be forwarded through the MPRs) to the MPRs. Then, the watchdog overhears the packets being forwarded by the MPRs and compares each overheard packet with the packets in the buffer to see if there is a match. If so, then the MPR has retransmitted the packet received from the source. This helps the watchdogs to fill two main attributes: *number of packets to be forwarded*, and *number of packets actually forwarded*. Thereafter, the watchdog compares these two parameters to have an initial idea on the classes of the MPRs. Thus, if *number of packets actually forwarded* $<$ *number of packets to be forwarded* for a certain MPR, then the watchdog marks this MPR as "malicious". Otherwise, the MPR is marked as "cooperative". The algorithm of the Data Collection phase that runs at each watchdog in the cluster is presented in Algorithm 1.

Algorithm 1 Data collection.

Initialization:
Let $CH(C)$ be the cluster-head of cluster C.
Let $MPRSet(C)$ be the set of elected MPRs in cluster C.
Let $WatchdogSet(C)$ be a the set of watchdogs in cluster C.
Let $PacketSet(m)$ be a the set of packets to be sent by the MPR m.
Let s be a packet sent by a source node to an MPR.
Let $countPacketsToSend(m)$ be the number of packets to be sent by the MPR m.
Let $countSentPackets(m)$ be the number of packets actually sent by the MPR m.
Let $countDroppedPackets(m)$ be the number of packets dropped by the MPR m.

```
procedure DataCollection
    for each MPR m ∈ MPRSet(C) do
        for each packet p ∈ PacketSet(m) do
            increment countPacketsToSend(m)
            if p = s then
                increment countSentPackets(m)
            else
                increment countDroppedPackets(m)
            end if
        end for
        if countSentPackets(m) < countPacketsToSend(m) then
            InitialClass(m) := malicious
        else
            InitialClass(m) := cooperative
        end if
    end for
end procedure
```

However, this decision is not final. In fact, the observations of single watchdogs are not enough to construct an accurate decision in such kind of networks. This is due to the fact that: (1) the high mobility of vehicles may hinder the monitoring of some watchdogs, (2) the packets collisions, which occur when two or more vehicles located within the same transmission range are sending to the same destination, may cause an ambiguity in the monitoring and leads hence to false evidences, and (3) some watchdogs are themselves malicious in the sense that they may mislead and claim falsified evidences. Here lies the importance of the data exchange, data analysis, and data propagation phases.

4.2. Data exchange

In this phase, the watchdogs located in the same cluster share their collected evidences. To do so, each watchdog broadcasts a *HELLO* message (Wahab et al., 2013b) that contains its own IP address to the other watchdogs to be able to exchange messages. Thereafter, these watchdogs use the broadcasting communication method implemented in the VANET QoS-OLSR protocol (Wahab et al., 2013b) to share their observations. In this way, each watchdog will have a database containing a representative set of evidences according to which it can build the final decisions. The algorithm depicting this phase running at each watchdog vehicle is given by Algorithm 2.

Algorithm 2 Data exchange.

1: **Initialization:**
2: Let $watchdogs(C)$ denote the set of watchdogs in cluster C.
3: Let w be a watchdog vehicle in $watchdogs(C)$ running this algorithm.
4: Let $Observations(x)$ be the set of observations collected by the watchdog x.

5: **procedure** DataExchange
6: **for each** watchdog $\bar{w} \in watchdogs(C) \mid \bar{w} \neq w$ **do**
7: Broadcast *HELLO* message to $\bar{w}$
8: Broadcast the observations set to $\bar{w}$, i.e., $Observations(\bar{w}) := Observations(\bar{w}) \cup Observations(w)$
9: **end for**
10: **end procedure**

4.3. Data analysis

In this phase, the Support Vector Machine (Han et al., 2012) classification technique is used to analyze the collected data and classify the MPRs. This phase is important to enhance the detection results by taking into account that some falsified evidences may affect the final decisions. Namely, some watchdogs may misjudge some MPRs either intentionally or unintentionally. These watchdogs may intentionally classify some cooperative MPRs as malicious in order to exclude them from the competition in the future election/selection procedures or classify other malicious MPRs as cooperative if an alliance between them occurs. Moreover, some watchdogs may unintentionally report false evidences as a result of the ambiguity in the monitoring caused either by packets collisions or by the high mobility of vehicles. The data analysis phase, which consists of analyzing the training set using SVM and classifying the MPRs accordingly, runs each t iteration and can be called for by the watchdogs to run out of the regular iterations (i.e., in the time between iteration t and iteration $t+1$) each time these watchdogs suspect a malicious behavior in the time between iterations.

In this phase, each watchdog splits its own database into two sets: training set and test set. The test set represents the observations of the watchdog itself, while the training set contains all other watchdogs' observations. The idea is that the watchdog considers its observations insufficient to make a decision. Therefore, it considers them as test set and waits for the other watchdogs' observations (training set) to analyze them and predict the class labels (cooperative/malicious) of its own set. Thus, the watchdog uses Support Vector Machine (SVM) for this purpose. For the sake of increasing the accuracy, SVM works in both incremental and online fashion. It is incremental in the sense that the training set is continuously populated with new training tuples representing the new evidences collected by the watchdogs from one iteration to another, while keeping the main set of training examples collected previously. It is worth to note that only the support vectors (those essential tuples that are located closest to the decision hyperplane that separates the tuples classified as cooperative from those classified as malicious and whose removal would change the position of the hyperplane and affect thus the whole classification process) are migrated throughout iterations (not the whole training set). As well, SVM is used in online manner so that the classes of the training set in each iteration are continuously updated by what is learned from the previous iterations. The algorithm of this phase running at each watchdog vehicle is shown in Algorithm 3.

Algorithm 3 Data analysis.

1: **Initialization:**
2: Let $watchdogs(C)$ denote the set of watchdogs in cluster C.
3: Let w be a watchdog vehicle in $watchdogs(C)$ running this algorithm.
4: Let $watchdogs(C) \setminus \{w\}$ denote the set of all watchdogs in $watchdogs(C)$ except for w.
5: Let *observations* be the observations collected by the watchdog w.
6: Let *train* be the training set of the watchdog w.
7: Let *test* be the test set of the watchdog w.
8: Let *SupportVectors* be the support vectors used by the classifier of watchdog w to distinguish among classes.
9: Let $MPRSet(C)$ denote the set of MPRs elected in cluster C.

10: **procedure** DataAnalysis
11: **for each** MPR $m \in MPRSet(C)$ **do**
12: $train := SupportVectors \cup observations(watchdogs(C) \setminus \{w\})$
13: $test := observations$
14: Train the classifier to find the hyperplane that maximizes the margin between classes in *train* by solving Eq. (7)
15: Learn the classifier C on *train* by pairing each set of inputs with the expected output
16: Use the learned classifier C to predict the final classes of *test*
17: **end for**
18: **end procedure**

SVM is a classification technique that employs a nonlinear mapping to convert the original data into higher dimension in order to find a hyperplane that separates the training tuples based on their classes. The hyperplane is determined using support vectors (major training records) and margins (determined based on the support vectors) in a way that maximizes the hyperplane's margins with the aim of delivering more accurate results when classifying future data tuples (Han et al., 2012).

Formally, the SVM technique works as follows. Let's define $(x_i, y_i)\ldots(x_n, y_n)$ to be the training data, n to be the number of observations, $y_i \in \{-1, +1\}$ being the class label such that -1 indicates a malicious vehicle and $+1$ indicates a cooperative vehicle, x_i being each observation, w being a weight vector and b a threshold such that $y_i(w \times x_i + b) \geq 1$. Note that $y_i(w \times x_i + b)$ represents the functional margin that tells whether a particular point is properly classified or not, where $w \times x_i + b$ is the model's prediction for the ith training tuple and y_i is the actual class label (i.e., cooperative or malicious). The larger is the functional margin, the better is the separation between classes and the more accurate are the classifications. If $y_i = 1$ (i.e., cooperative), we need $w \times x_i + b$ to be a large positive number for a large functional margin. If $y_i = -1$ (i.e., malicious), we need $w \times x_i + b$ to be a large negative number for a large functional margin. This is because $y_i(w \times x_i + b) >= 0$ means that the classification is correct. To prevent data points from falling

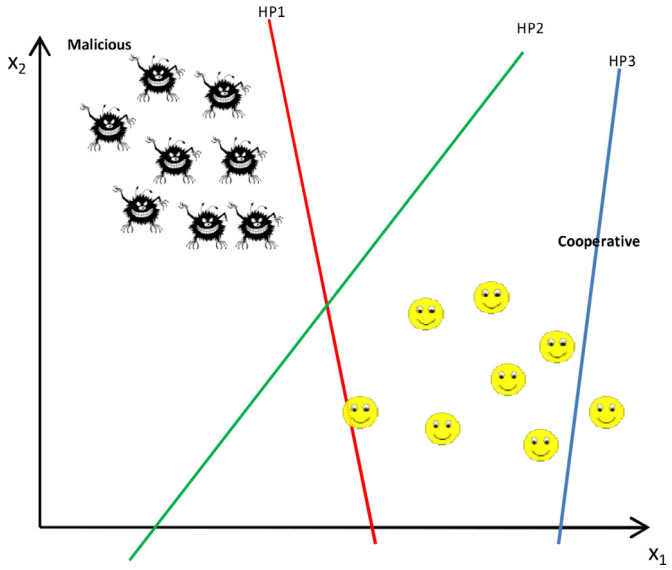


Fig. 4. According to SVM, the hyperplane HP2 is chosen to separate the two classes since it gives the maximum margin between them. HP1 separates the classes but does not maximize the margin between them, while HP3 cannot separate the classes.

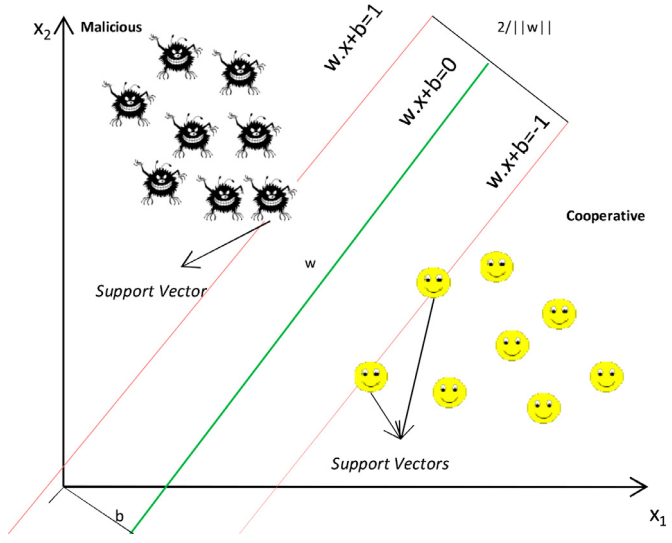


Fig. 5. The value of w affects the position of the hyperplane.

into the margin when determining the optimal hyperplane (i.e., to maximize the functional margin), the following constraints have to be satisfied:

- $w \times x_i + b >= 1$ for $x_i = 1$ and
- $w \times x_i + b <= -1$ for $x_i = -1$

These constraints can be combined into one set of inequalities as follows:

$$y_i(w \times x_i + b) \geq 1 \quad for \quad all \quad 1 \leq i \leq n. \tag{2}$$

Each MPR can be either cooperative or malicious. Thus, we define $y_i \in \{cooperative, malicious\}$. Given the two classes, the main idea of SVM is to maximize the margin between these classes (Fig. 4). Given the training datasets $(x_i, y_i)\ldots(x_n, y_n)$, the objective is to find the hyperplane that have a maximum margin (Fig. 5) such that:

$$w \times x + b = 0 \tag{3}$$

Thus, the training observations satisfy:

$$w \times x + b \geq -1 \quad for \quad all \quad malicious \quad x_i \tag{4}$$

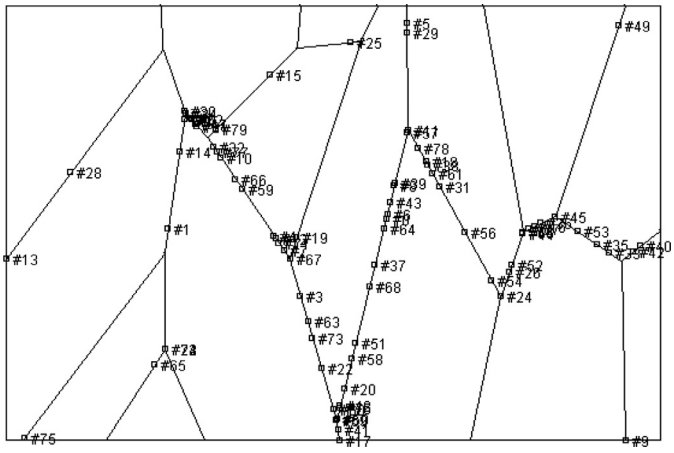


Fig. 6. The area used to simulate the movement of vehicles in VanetMobiSim.

$$w \times x + b \leq +1 \quad for \quad all \quad cooperative \quad x_i \tag{5}$$

The problem of finding the optimal hyperplane can be turned into a convex optimization problem Scholkopf and Smola (2001):

$$\begin{cases} min\left\{\frac{\|w\|^2}{2} + c\sum_{i=1}^{n}\varepsilon_i\right\} \\ Subject\ to\ \ y_i(w \times x_i + b) \geq 1 - \varepsilon_i, \varepsilon_i \geq 0, 1 \leq i \leq n \end{cases} \tag{6}$$

$\sum_{i=1}^{n}\varepsilon_i$ is used to relax the constraints on the learning vectors, and c represents a constant that is responsible for achieving the trade-off between maximizing the margin and minimizing the number of misclassifications. The convex optimization problem presented in Eq. 6 can be solved by using the Lagrange multiplier Scholkopf and Smola (2001):

$$\begin{cases} maximize\ \ L(\alpha) = \sum_{i=1}^{n}\alpha_i - \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n}\alpha_i\alpha_j y_i y_j K(x_j, x_i) \\ subject\ \ to\ \ \sum_{i=1}^{n} y_i\alpha_i = 0,\ \ and\ \ 0 \leq \alpha_i \leq C\ for\ \ all\ \ 1 \leq i \leq n \end{cases} \tag{7}$$

where α_i are the Lagrange multipliers and $K(x_j, x_i)$ represents the kernel function. There are four main kernel functions Han et al. (2012) used for SVM: linear, polynomial, gaussian, and sigmoid.

$$K(x_j, x_i) = \begin{cases} x_i.x_j, linear \\ (\gamma.x_i.x_j + c)^d, polynomial \\ \exp(-\gamma.|\ x_i - x_j\ |^2), gaussian\ radial\ basis \\ \tanh(\gamma.x_i.x_j + c), sigmoid \end{cases} \tag{8}$$

By solving Eq. 7, we should get (Scholkopf & Smola, 2001):

$$w = \sum_{i=1}^{n}\alpha_i y_i x_i \tag{9}$$

Finally, the decision function is given by:

$$f(x, \alpha, b) = \{\pm 1\} = sgn\left(\sum_{i=1}^{n} y_i\alpha_i K(x_j, x_i) + b\right) \tag{10}$$

4.4. Data propagation

In order to mitigate the detection time and overhead, the data propagation phase is proposed. After building the final decisions and classifying the MPRs, the cluster-heads may exchange the classes of the MPRs along with the routing information. Note that this phase is an optional phase that happens only whenever there exists a path composed of cooperative MPRs between cluster-heads. The idea is to allow the cluster-heads to exchange the classification results between each other in order to reduce the overhead of either exchanging results between each set of vehicles or

repeating the detection mechanism steps for the already detected malicious vehicles. Thus, our proposed detection model can still effectively work even when all the paths between clusters are composed of malicious MPRs. Each cluster-head is then responsible for broadcasting these classes to their cluster members. The idea is to make the larger possible set of vehicles aware of the MPR nodes that are already classified as malicious so that these vehicles will refrain from cooperating with them or electing/selecting them if they fall later within the range of their clusters. Thus, we are reducing the detection time, the computation overhead, and the communication overhead. Practically, the vehicles will know the malicious MPRs without initiating new detection mechanisms over and over again knowing that the detection requires time, computation overhead caused by applying the SVM algorithm, and communication overhead caused by the exchange of messages during the data exchange phase. However, the vehicles will still monitor the vehicles that are classified cooperative since some selectively malicious vehicles may well-behave in one cluster and then misbehave in other clusters. The algorithm of the data propagation phase is explained in Algorithm 4.

Algorithm 4 Data propagation.

Initialization:
Let $C1$ and $C2$ be two different clusters.
Let x be a cluster head of cluster $C1$.
Let y be a cluster head of cluster $C2$.
Let $members(x)$ be the members in the cluster leaded by the cluster-head x.
Let $members(y)$ be the members in the cluster leaded by the cluster-head y.
Let $maliciousSet(x)$ be the set of vehicles classified as malicious within $C1$.
Let $maliciousSet(y)$ be the set of vehicles classified as malicious within $C2$.
Let $cooperativeSet(x)$ be the set of vehicles classified as cooperative within $C1$.
Let $cooperativeSet(y)$ be the set of vehicles classified as cooperative within $C2$.

procedure DATA PROPAGATION
  if a contact between x and y happens **then**
    $maliciousSet(y) := maliciousSet(y) \cup maliciousSet(x)$
    $maliciousSet(x) := maliciousSet(x) \cup maliciousSet(y)$
    $cooperativeSet(y) := cooperativeSet(y) \cup cooperativeSet(x)$
    $cooperativeSet(x) := cooperativeSet(x) \cup cooperativeSet(y)$
  end if
  for each vehicle $i \in$ members(x) **then**
    $maliciousSet(i) := maliciousSet(i) \cup maliciousSet(x)$
    $cooperativeSet(i) := cooperativeSet(i) \cup cooperativeSet(x)$
  end for
  for each vehicle $j \in$ members(y) **then**
    $maliciousSet(j) := maliciousSet(j) \cup maliciousSet(y)$
    $cooperativeSet(j) := cooperativeSet(j) \cup cooperativeSet(y)$
  end for
end procedure

4.5. Complexity analysis

Despite its effectiveness and ability to produce very accurate classifiers, the efficiency of SVM is a serious challenge. Practically, the complexity of SVM training is $O(n^3)$ in terms of time and $O(n^2)$ in terms of space, where n represents the training set size (Tsang, Kwok, & Cheung, 2005). Thus, it becomes infeasible for very large data sets. Unlike the existing SVM-based detection techniques in wireless networks Flouri et al. (2008); Kaplantzis et al. (2007); Li et al. (2010); Sedjelmaci and Feham (2011), we account for this fact and reduce the training set size n by (1) adapting SVM to a cluster-based architecture where vehicles have to store and analyse the data concerning their cluster members solely; (2) restricting the collection, storage, and analysis of data to concern exclusively a set of specialized nodes (i.e., MPRs) whose number tends to be small (Wahab et al., 2013b); (3) exploiting the fact that SVM uses only the support vectors for classification and assume that only these support vectors amongst the other training records should be migrated from one iteration to another; and (4) proposing a propagation algorithm to disseminate the final decisions among clusters when exchanging routing information.

As for the time complexity of our model, Algorithm 1 runs at each watchdog that has to monitor the packets being sent by the MPRs belonging to the same cluster. Therefore, the time complexity of Algorithm 1 is $O(Pa)$, where Pa is the number of packets sent by these MPRs in the cluster. In Algorithm 2, each watchdog has to broadcast its set of collected observations to the other watchdogs of the same cluster. Thus, the time complexity of Algorithm 2 is $O(1)$. As for Algorithm 3, each watchdog w has to run SVM in order to analyze the training set consisting of the observations of the other cluster C's watchdogs, i.e., $watchdogs(C)\backslash\{w\}$. Thus, the running time of Algorithm 3 is $O(M \times n^3)$, where M is the number of MPRs, n is the number of observations of the $watchdogs(C)\backslash\{w\}$ watchdogs, and $O(n^3)$ is the cost of applying SVM on the training set of size n. As the number of MPRs is fixed (usually small), the running time of Algorithm 3 is $O(n^3)$. Finally, Algorithm 4 involves propagating final decisions among cluster-heads whenever a contact occurs. Therefore, the running time of Algorithm 4 is $O(1)$. Obviously, the main complexity lies in Algorithm 1 (i.e., $O(Pa)$) and Algorithm 3 (i.e., $O(n^3)$). Clearly, the number of observations concerning the exchanged packets is greater than the number of packets itself since the observations are being collected from several watchdogs, i.e., $n > Pa$. Therefore, the overall time complexity of CEAP is $O(n^3)$. Recall that n in our model is small enough so that the polynomial complexity of CEAP is practically not a drawback.

As for the communication overhead, a combination of TESLA Perrig, Canetti, Tygar, and Song (2005) and Public Key Infrastructure (PKI) Gura, Patel, Wander, Eberle, and Shantz (2004) can be used as an authentication protocol for inter-vehicle communications. In fact, these two techniques have proved to be lightweight for resource-constrained nodes such as vehicles in VANET. In practice, verifying a signature using PKI requires 0.43s Gura et al. (2004). It is worth noting as well that the use of this combination allows us to satisfy three main security requirements; namely, integrity, authentication, and freshness Wahab et al. (2013b).

5. Simulation scenario and experimental results

In this section, we explain the scenario and parameters used to perform our simulations. Then, we present empirical results obtained after comparing our CEAP model using different kernel functions as a first step and with the original SVM, averaging, and Dempster–Shafer models as a second step. (Flouri et al., 2008; Kaplantzis et al., 2007; Kargl, Klenk, Weber, & Schlott, 2004; Li et al., 2010; Sedjelmaci & Feham, 2011; Wahab et al., 2013a, 2014).

5.1. Simulation scenario and parameters

To perform the simulations, VanetMobiSim (Fiore, Harri, Filali, & Bonnet, 2007) has been used as a road traffic simulator and MATLAB Gilat (2008) as a means to implement the network-related

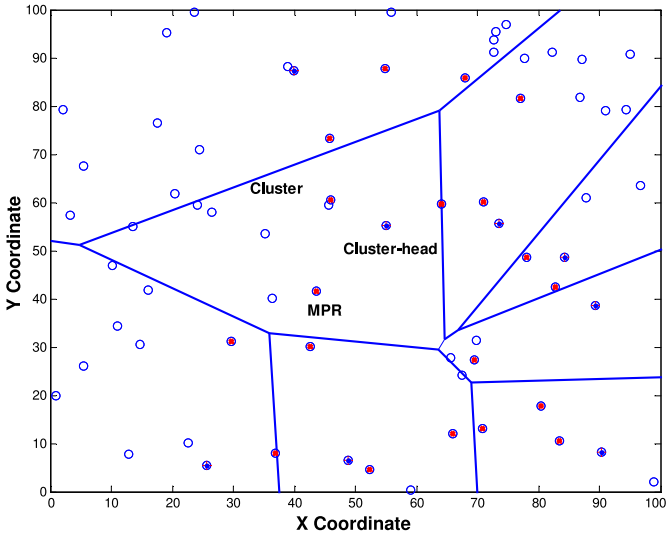


Fig. 7. VANET QoS-OLSR clustering example.

algorithms. We have successfully used this combination of simulators in several works before (e.g., Wahab et al. (2013a, 2013b, 2014)). VanetMobiSim is an XML-based traffic simulator that enables the user to determine the vehicular network features such as topography, velocity, number of nodes, time steps, and duration. It advances a platform to execute all the steps of single mobility simulation. The choice of VanetMobisim stems from its ability to provide excellent trace support for vehicular mobility simulations compared to other vehicular mobility generators and generate realistic mobility patterns (Härri, Filali, Bonnet, & Fiore, 2006; Narendra & Savita, 2014; Shea, Hassanabadi, & Valaee, 2009). A simulation area of 5000 × 5000 m is used to simulate a set of vehicles varying from 100 to 500. The screenshot of this area is presented in Fig. 7. A multi-directional multi-lane highway topology is used to simulate the traffic. The minimum allowed speed on this highway was set to 60 km/h, while the maximum speed was 120 km/h. The vehicles attain their maximum speed during the main highway pass, and then slow considerably when they reach the turns, which produces a realistic model with both low and high density traffic. Each simulation round lasted 420 s (i.e., 7 min) during which vehicles are in a continuous movement so that new vehicles may enter and existing vehicles may leave the area. However, the fact that we simulate a large number of vehicles (up to 500 vehicles) makes the area always populated with some vehicles during the simulation time. After the simulation has been completed, VanetMobisim generates a file containing some important features such as time, velocity, and position. These parameters are used to populate the algorithms developed using MATLAB. The transmission range used to simulate the network is 300 m (Kaur, Kaur, & Singh, 2012). The percentage of malicious vehicles varies from 10% to 50%. The code considers as well the cases of packet collisions and untrustworthy watchdogs. We assume in our simulations that 20% of the watchdogs are untrustworthy and provide falsified evidences. The packet drop percentage varies from 1% to 100% so that a malicious vehicle can either drop all the packets or a portion of them.

After running the code in MATLAB, we obtain a database with six features: $time$, $watchdog-id$, $MPR-id$, $num-packets-to-forward$, $num-packets-actually-forwarded$, and $class-MPR$. For example, the following entry: 10, 1, 3, 20, 15, "malicious" can be read as follows. At second 10 of simulations, the watchdog referenced by 1 monitored the MPR vehicle referenced by 3, which is supposed to forward 20 packets. The watchdog found that the MPR 1 has forwarded only 15 packets out of 20; therefore this watchdog decided to classify the MPR 1 as malicious initially awaiting for other evidences coming from the other watchdogs. Thereafter, we apply the SVM algorithm on this database using MATLAB as well. To this end, we split the database into *training set* (Table 4) and *test set* using *K*-fold cross-validation with $k = number\ of\ watchdogs$. Thus, if the number of watchdogs is 50, there will be 50 iterations with different test set each (i.e, each watchdog's observations will be a test set once). Thereafter, we train the SVM on the training set, predict the class labels of the test set, and then compare the predicted classes with the actual classes to evaluate the performance of our model by means of confusion matrix (Han et al., 2012). It is worth mentioning as well that the Ant-Colony-Optimization-based routing algorithm developed in VANET QoS-OLSR (Wahab et al., 2013b) is used to simulate the routing process.

Table 3
Simulation parameters.

Parameter	Value
Number of iterations	10,000 (95% of confidence level)
Simulation area	5000 × 5000 m
Number of vehicles	100, 200, 300, 400, and 500
Percentage of malicious vehicles	10%, 20%, 30%, 40%, and 50%
Transmission range	250 m
Topology	Multi-lane multi-directional highway
Packet Size	1 kb
Percentage of untrustworthy watchdogs	20%
Minimum speed	60 km/h
Maximum speed	120 km/h
Packet drop percentage	from 1% to 100%
Cross-validation	*K*-fold with $k = number\ of\ watchdogs$
Kernel functions	linear, polynomial, multilayer, quadratic, and gaussian

Note that, we apply the SVM on the training set using five different kernel functions (linear, polynomial, multilayer, quadratic, and gaussian) in order to choose the function that best fits our model. We compare also the winner kernel function with the averaging-based, Dempster–Shafer-based, and original SVM-based detection models to show how CEAP is able to enhance the detections. The parameters (Chen & Chen, 2008) used to evaluate the performance of the different models are the accuracy rate, attack detection rate, false positive rate, and packet delivery ratio.

$$Accuracy\ Rate = 100\% \times \frac{Total\ number\ of\ correctly\ classified\ processes}{Total\ number\ of\ processes} \quad (11)$$

$$Attack\ Detection\ Rate = 100\% \times \frac{Total\ number\ of\ attacks}{Total\ number\ of\ detected\ attacks} \quad (12)$$

$$False\ Positive\ Rate = 100\% \times \frac{Total\ number\ of\ misclassified\ processes}{Total\ number\ of\ normal\ processes} \quad (13)$$

$$Packet\ Delivery\ Ratio = \frac{Total\ number\ of\ recieved\ packets}{Total\ number\ of\ sent\ packets} \quad (14)$$

For the sake of increasing the accuracy of our simulations, we considered a confidence level of 95%. Then, we run independent simulations for each factor being evaluated (i.e., accuracy, attack detection rate, false positive, and packet delivery ratio) and calculate the confidence interval using mean and standard deviation metrics to learn the number of simulation runs that are able to yield results respecting this interval. Experiment results show that running 10,000 independent iterations is able to provide results that fall within the expected confidence interval. The simulation parameters are summarized in Table 3.

Table 4
Example of training set.

Time	Watchdog	MPR	Num-packets-to-forward	Num-packets-actually-forwarded	class-MPR
10	1	4	20	20	"cooperative"
10	1	5	20	13	"malicious"
10	2	5	20	12	"malicious"
13	3	5	15	0	"malicious"
14	3	10	35	35	"cooperative"
14	1	4	25	24	"malicious"
…	…	…	…	…	…

5.2. Simulation results

In order to evaluate the performance of our proposed detection model, we first compare it using different kernel functions in order to select the one that best fits our model. Thereafter, we compare the winner kernel function with the averaging-based Kargl et al. (2004), Dempster–Shafer-based Chen and Venkataramanan (2005); Konorski and Orlikowski (2009); Li and Joshi (2009); Wahab et al. (2013a, 2014), and original SVM-based Flouri et al. (2008); Kaplantzis et al. (2007); Li et al. (2010); Sedjelmaci and Feham (2011) models, which are widely used in the literature for intrusion detection in wireless networks. In the averaging model, nodes offer a number between 0 and 1 to vote on a node's misbehavior and the decision is the average of these votes. In the Dempster–Shafer models, each evidence is assigned a weight equal to the trustworthiness level of the node giving the evidence and the decision is made after aggregating these evidences to come up with a degree of belief Schubert (2011). SVM is a classification technique that employs a nonlinear mapping to convert the original data into higher dimension in order to find a hyperplane that best separates the training tuples based on their classes (Han et al., 2012). Our model uses SVM in incremental manner such that the training set is continuously accumulated with new evidences. In addition, SVM is performed in online fashion so that the classes of the training set in each iteration are continuously updated by what is learned from the previous iterations. The simulations are done according to two scenarios: (1) different number of vehicles, (2) and different percentage of malicious vehicles. This helps study the scalability of our model against the increase in both the network density and percentage of malicious vehicles. In the first scenario, the percentage of malicious vehicles was set to 30%, while this percentage varies in the second scenario (10%, 15%, 20%, 25%, 30%, 35%,40%, 45%, and 50%). The number of vehicles in the second scenario was set to 500, while this number is variable in the first scenario (100, 150, 200, 250, 300, 350, 400, 450, 500).

We observe from Tables 5 and 6 that the Gaussian Radial Basis Function kernel outperforms narrowly the other kernel functions in terms of accuracy, attack detection, and false positive rates in both scenarios. The Gaussian Radial Basis Function kernel, which is most widely used for SVMs classification except for text recognition applications, adds a bump around each data point and supposes that each kernel entry is a dissimilarity measure computed as a square of Euclidean distance between two data points in a negative exponential manner (Han, Embrechts, & Szymanski, 2011; Schlkopf et al., 1997). Note that the values in Table 5 are obtained after running the simulations for each kernel function using different number of vehicles (100, 200, 300, 400, and 500) according to each performance metric (accuracy rate, attack detection rate, and false positive rate) and computing the average accordingly. Similarly, the values in Table 6 are obtained after running the simulations for each kernel function using different percentages of malicious vehicles (10%, 20%, 30%, 40%, and 50%) according to each performance metric (accuracy rate, attack detection rate, and false positive rate) and computing the average accordingly. The number of iterations in both scenarios is 10000.

In Figs. 8, 9, and 10, we compare the winner kernel function of SVM in CEAP, Gaussian Radial Basis Function kernel, with the averaging-based, Dempster–Shafer-based, and original SVM-based models. From Figs. 8a, 9a, and 10a, we remark that the SVM-based models (CEAP and original SVM) yield a better accuracy, attack detection, and false positive rates in the denser networks. This is the case because the bigger the number of involved vehicles is, the more evidences can be collected. This leads to have a more accurate view of misbehaviors.

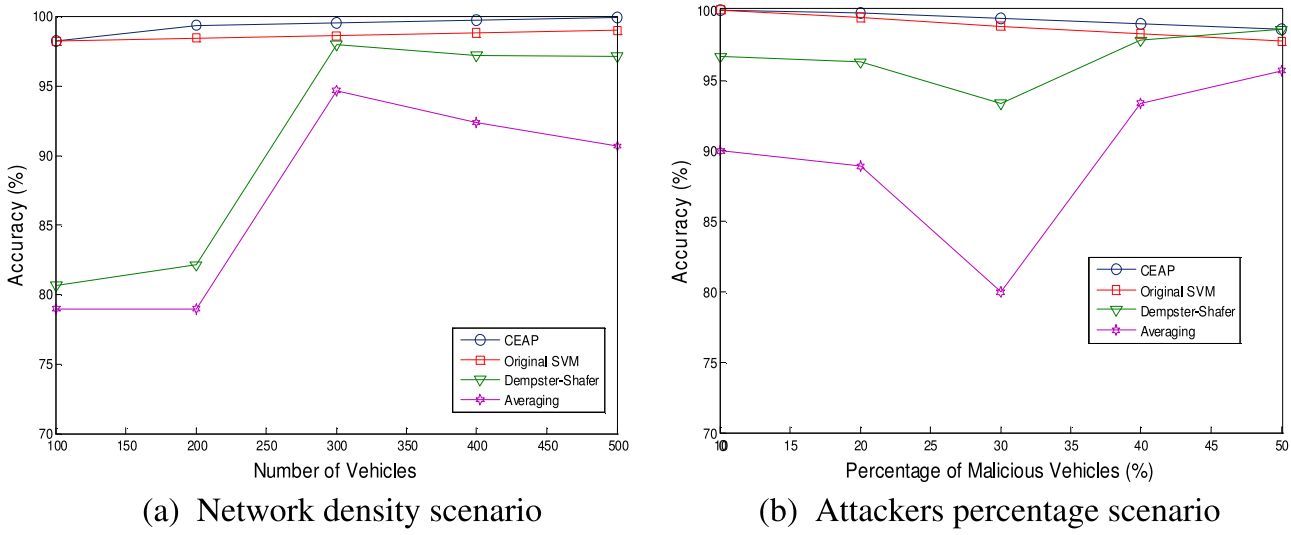
However, this is not always the case for the averaging and Dempster–Shafer models. Practically, the performance of the averaging model is affected by the number of outliers (misleading evidences) and not by the network density. Similarly, the performance of Dempster–Shafer is affected by the frequency of cumulative beliefs (the frequency of observations coming from a single watchdog), which makes both averaging and Dempster–Shafer instable in both scenarios (network density and percentage of malicious vehicles). For the percentage of malicious vehicles scenario, the performance of the SVM-based models is slightly decreased with the increase in the percentage of malicious vehicles as shown in Figs. 8b, 9b, and 10b.

Table 5
Kernel functions comparison according to the network density scenario.

Kernel function	Performance metric		
	Accuracy rate (%)	Attack detection rate (%)	False positive rate (%)
Linear kernel	99.1375	98.9175	1.0825
Multilayer percepton kernel	99.0452	98.5975	1.4025
Quadratic kernel	99.1375	98.9175	1.0825
Polynomial kernel	99.5375	99.1275	0.8725
Gaussian radial basis function kernel	99.6753	99.1575	0.8425

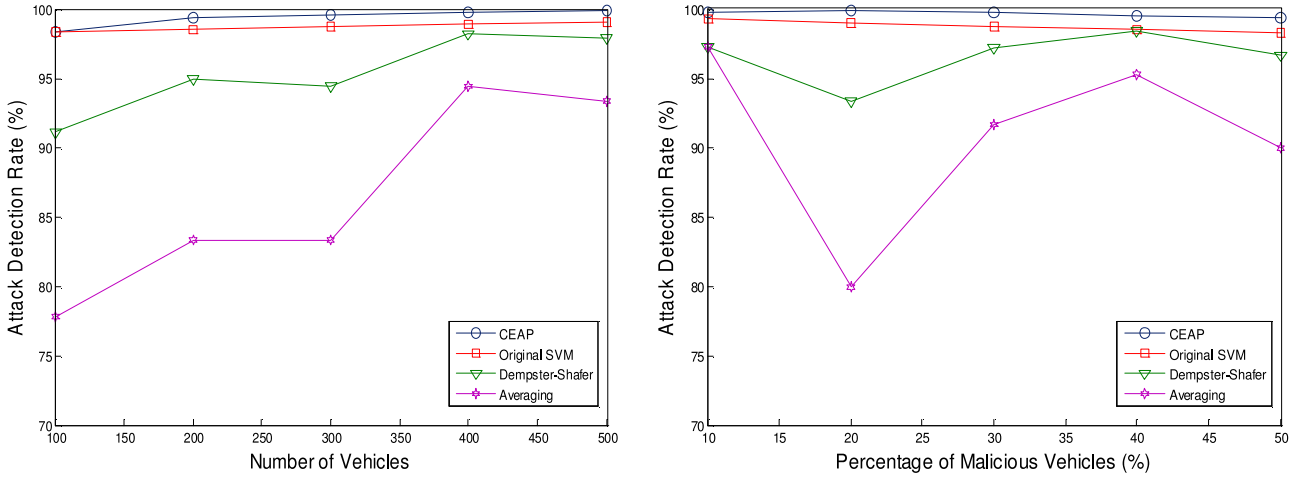
Table 6
Kernel functions comparison according to the attackers percentage scenario.

Kernel function	Performance metric		
	Accuracy rate (%)	Attack detection rate (%)	False positive rate (%)
Linear kernel	98.0500	98.0960	1.9040
Multilayer percepton kernel	97.9900	98	2
Quadratic kernel	98.0500	98.0960	1.9040
Polynomial kernel	98.3470	98.1900	1.8100
Gaussian radial basis function kernel	98.7220	98.9800	1.0200



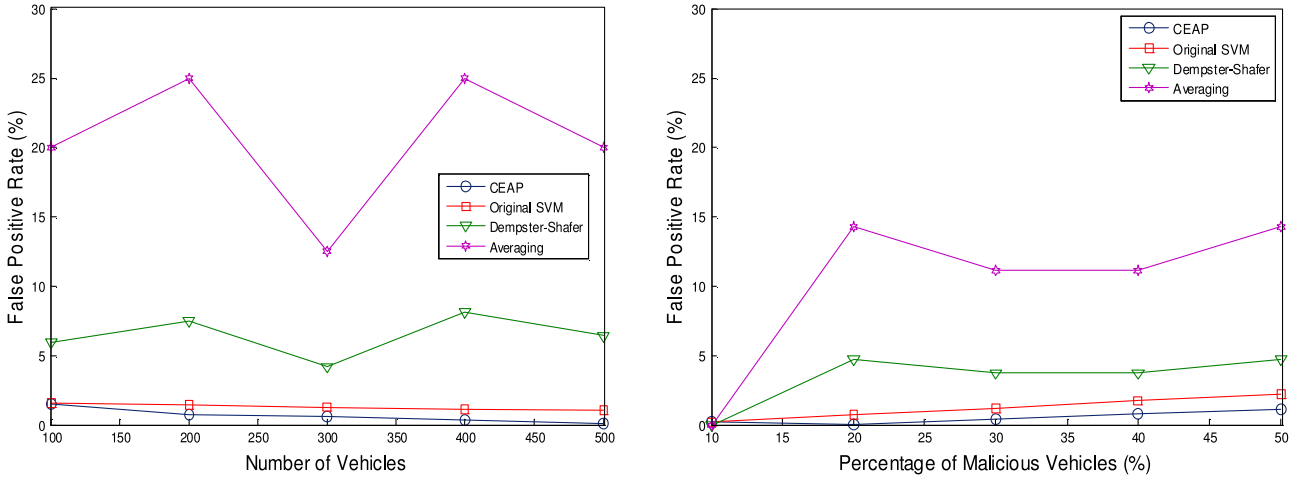
(a) Network density scenario (b) Attackers percentage scenario

Fig. 8. Accuracy rate comparison: CEAP, original SVM, Dempster–Shafer, and averaging models.



(a) Network density scenario (b) Attackers percentage scenario

Fig. 9. Attack detection rate comparison: CEAP, original SVM, Dempster–Shafer, and averaging models.



(a) Network density scenario (b) Attackers percentage scenario

Fig. 10. False positive rate comparison: CEAP, original SVM, Dempster–Shafer, and averaging models.

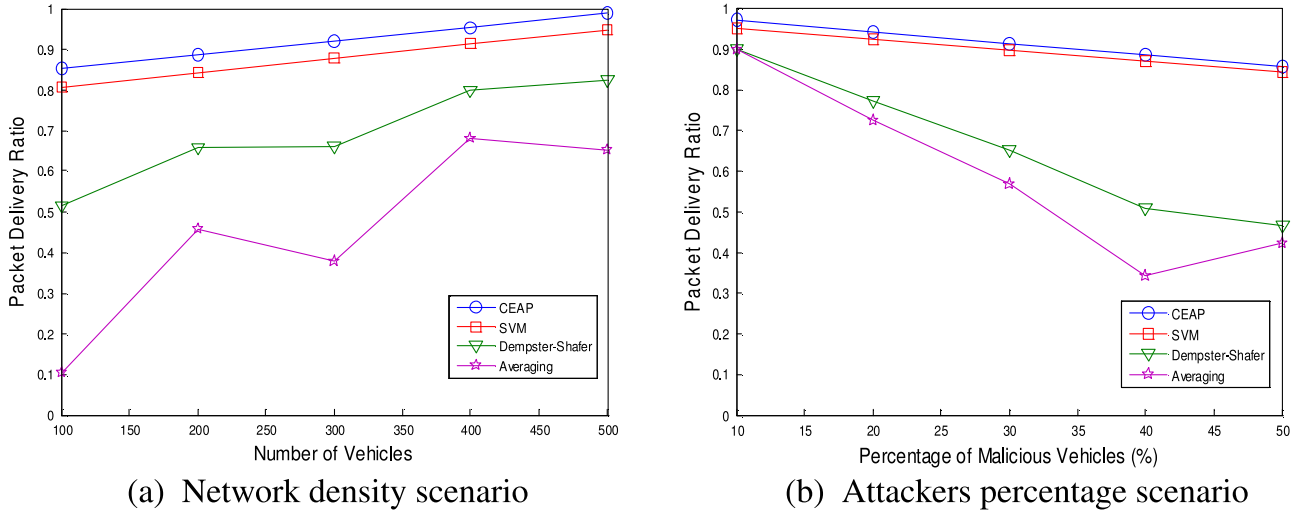


Fig. 11. Packet delivery ratio comparison: CEAP, original SVM, Dempster–Shafer, and averaging models.

Nevertheless, SVM-based models are still far more resilient to a significantly larger percentage of malicious vehicles in the different parameters (accuracy, attack detection, and false positive rates). Moreover, Figs. 8, 9, and 10 show that the SVM-based models outperform the averaging and Dempster–Shafer models in the different performance metrics (accuracy, attack detection, and false positive rates). This is due to the fact that the compromised watchdogs, in the averaging model, could cause an error in the final judgment by offering deliberately incorrect evidences (Chen & Venkataramanan, 2005), which justifies the instability in the averaging model plots. Additionally, the performance of the Dempster–Shafer is affected by the cumulative beliefs as proven by Josang and Pope (2012), which makes it inefficient to deal with such problem where multiple evidences are generated from each single watchdog. On the other hand, compromised evidences do not have a significant influence to the performance of SVM since it is resilient to the overfitting (Han et al., 2012; Pham & Triantaphyllou, 2008). In fact, SVM finds the hyperplane that maximizes the margin between the two classes and is able hence to accurately differentiate between these classes. Once a hyperplane is found, most of the data other than the support vectors (the closest points to the boundary) become redundant. Thus, making small changes to the data cannot significantly influence the hyperplane and hence the SVM. Therefore, Support Vector Machines are able to generalize very well. In addition, Figs. 8, 9, and 10 reveal that CEAP outperforms the original SVM-based models in terms of accuracy, attack detection, and false positive rates respectively. This is due to the fact that the training set in CEAP is accumulated in an incremental manner, which enriches the learning space of the SVM classifier and enhances its capability to differentiate among classes without re-training from scratch. Moreover, the classes of the training set in CEAP are updated in online fashion by what is learned from previous iterations, which enhances the precision of the classifications as the training set based on which the classifier is learned is continuously becoming more and more accurate. As a result, CEAP is able to enhance the packet delivery ratio as shown in Fig. 11. This is due the fact that CEAP minimizes the effect of malicious vehicles on the network by accurately detecting and identifying them, which leads to improve the routing process consequently.

6. Conclusion and discussions

This work addresses the problem of detecting malicious vehicles in clustered VANETs. To this end, a cluster-based cooperative detection model, called CEAP, is advocated. In CEAP, the cluster members are designated as watchdogs to monitor and collect evidences on the behavior of the Multi-point relay (MPR) nodes that are responsible for forwarding the packets on behalf of the cluster. Thereafter, the SVM learning technique is employed to classify MPRs either cooperative or malicious. CEAP enjoys three main advantages over the existing detection mechanisms in VANETs: (1) it is able to increase the detection rate and minimize the false alarms by using SVM in incremental and online fashions; (2) it is able to adapt and effectively operate under the highly mobile nature of vehicles; and (3) it minimizes the overhead by reducing the size of the analyzed training dataset. Different kernel functions of SVM are simulated to select the function that best fits our model. Simulation results show that the Gaussian Radial Basis Function kernel slightly outperforms the other functions in terms of accuracy, attack detection, and false positive rates.

Promisingly, this work gives guidance to an intelligent detection model that can improve the detection rate and minimize the false alarms under highly mobile settings and with minimal overhead. The main novelty of CEAP w.r.t the state-of-the-art intelligent detection models (Abadeh, Mohamadi, & Habibi, 2011; Depren, Topallar, Anarim, & Ciliz, 2005; Wang, Yan, Wang, & Liu, 2011; Yi, Wu, & Xu, 2011) is its consideration for the real challenges of the environment within which it should operate. Specifically, our model is able to operate and achieve high detection rates in highly dynamic environments such as VANET. Moreover, CEAP differs from the state-of-the-art SVM-based detection models by reducing the training set size and that constitutes a real obstacle against the adoption of SVM in many domain applications despite its well-known effectiveness in terms of high detection rates. In order to verify its applicability in real environments, CEAP is tested using s realistic and highly mobile simulation environment implemented using the well-known VANET simulator VanetMobisim and under two different challenging scenarios: (1) variation in the network nodes density, and (2) increase in the percentage of malicious nodes. Simulation results prove that CEAP is quite resilient to both cases in the sense that it can perform well in low and high nodes' density and is not significantly influenced by the increase in the number of malicious attackers. Moreover, the simulation results reveal that CEAP outperforms the classical SVM-based, Dempster–Shafer-based, and averaging-based detection techniques in terms of accuracy, attack detection rate, false positive rate, and packet delivery ratio in both studied scenarios.

However, the main limitation of the proposed model is its reliance on a semi-honest adversary model whereby the cluster-head

is assumed to be a trusted third party. Therefore, it is worth investigating in the future a more complicated detection model wherein all the involved parties, including the cluster-heads, MPRs and normal nodes, are assumed to behave maliciously. Such a scenario requires a fully distributed model that does not depend on the behavior of any involved entity. Moreover, CEAP may be extended to study more complicated detection scenarios. For example, some malicious MPRs might send the packets to some vehicles other than the intended destinations instead of dropping them. This requires an advanced and sophisticated monitoring mechanism. In addition, the model could be extended to consider an intelligent multi-class (not binary) classification model that sub-classifies the malicious nodes based on the gravity and/or type of their attacks. This model should be able to tell whether a particular node is sinkhole attacker, DoS attacker, packet dropper, etc. Another future direction is the actions that should be taken after the detection. For example, such a model should consider rewarding the cooperative nodes and punishing the malicious ones in such a thoughtful manner that motivates the well-behavior and demotivates the misbehavior. This should be done while accounting for the decentralized and infrastructureless nature of VANET that plays against the existence of a trusted third party to take such responsibilities.

Acknowledgment

This work was supported by the Associated Research Unit of the National Council for Scientific Research CNRS Lebanon, Lebanese American University (LAU), Fonds de Recherche du Québec - Nature et Technologie (FRQNT), Khalifa University of Science, Technology & Research (KUSTAR), NSERC (Canada) and FQRSC (Québec).

References

Aad, I., Hubaux, J.-P., & Knightly, E.-W. (2008). Impact of denial of service attacks on ad hoc networks. *IEEE/ACM Transactions on Networking, 16*(4), 791–802.

Abadeh, M. S., Mohamadi, H., & Habibi, J. (2011). Design and analysis of genetic fuzzy systems for intrusion detection in computer networks. *Expert Systems with Applications, 38*(6), 7067–7075.

Auria, L., & Moro, R.-A. (2008). Support vector machines (SVM) as a technique for solvency analysis. *Technical Report 811*. DIW Berlin, German Institute for Economic Research.

Buchegger, S., & Boudec, J.-Y. L. (2002). Performance analysis of the confidant protocol. In *Proceedings of the 3rd acm international symposium on mobile ad hoc networking & computing* (pp. 226–236).

Chen, R.-C., & Chen, S.-P. (2008). Intrusion detection using a hybrid support vector machine based on entropy and TF-IDF. *International Journal of Innovative Computing, Information and Control (IJICIC), 4*(2), 413–424.

Chen, T.-M., & Venkataramanan, V. (2005). Dempster-Shafer theory for intrusion detection in ad hoc networks. *IEEE Internet Computing, 9*(6), 35–41.

Cheng, H., Yang, S., & Cao, J. (2013). Dynamic genetic algorithms for the dynamic load balanced clustering problem in mobile ad hoc networks. *Expert Systems with Applications, 40*(4), 1381–1392.

Clausen, T., Jacquet, P., Laouiti, A., Qayyum, A., & Viennot, L. (2002). Optimized link state routing protocol (OLSR). In *Proceedings of the multi topic conference (international)* (pp. 62–68). RFC Editor.

Daza, V., Domingo-Ferrer, J., Sebe, F., & Viejo, A. (2009). Trustworthy privacy-preserving car-generated announcements in vehicular ad hoc networks. *IEEE Transactions on Vehicular Technology, 58*, 1876–1886.

Depren, O., Topallar, M., Anarim, E., & Ciliz, M. K. (2005). An intelligent intrusion detection system (ids) for anomaly and misuse detection in computer networks. *Expert systems with Applications, 29*(4), 713–722.

Dorigo, M., Caro, G. D., & Gambardella, L.-M. (1999). Ant algorithms for discrete optimization. *Artificial Life, 5*(2), 137–172.

Douceur, J. R., & Moscibroda, T. (2007). Lottery trees: motivational deployment of networked systems. In *Proceedings of the conference on applications, technologies, architectures, and protocols for computer communications* (pp. 121–132).

Engoulou, R. G., Bellaïche, M., Pierre, S., & Quintero, A. (2014). VANET security surveys. *Computer Communications, 44*(15), 1–13.

Fiore, M., Harri, J., Filali, F., & Bonnet, C. (2007). Vehicular mobility simulation for VANETs. *40th Annual Simulation Symposium ANSS07, 07*, 301–309.

Flouri, K., lozano, B.-B. Tsakalides, P. (2008). Distributed consensus algorithms for SVM training in wireless sensor networks.

Gantsou, D. (2015). On the use of security analytics for attack detection in vehicular ad hoc networks. In *Proceedings of the international conference on Cyber security of smart cities, industrial control system and communications (ssic), 2015* (pp. 1–6). IEEE.

Gilat, A. (2008). *MATLAB: an introduction with applications*. Wiley.

Gura, N., Patel, A., Wander, A., Eberle, H., & Shantz, S. C. (2004). Comparing elliptic curve cryptography and rsa on 8-bit cpus. In *Proceedings of the cryptographic hardware and embedded systems-ches 2004* (pp. 119–132). Springer.

Han, J., Kamber, M., Pei, J., & Kaufmann, M. (2012). *Data mining: concepts and techniques* (3rd). San Francisco, CA, USA: The Morgan Kaufmann Series in Data Management Systems.

Han, L., Embrechts, M.-J., & Szymanski, B. (2011). Sigma tuning of gaussian kernels: detection of Ischemia from magnetocardiograms. *Computational Modeling and Simulation of Intellect: Current State and Future Perspectives*, (1), 206–223.

Härri, J., Filali, F., Bonnet, C., & Fiore, M. (2006). VanetMobiSim: generating realistic mobility patterns for VANETs, *VANET '06. Proceedings of the 3rd international workshop on vehicular ad hoc networks* (pp. 96–97). ACM.

Haykin, S. (1998). *Neural networks: a comprehensive foundation* (2nd). Upper Saddle River, NJ, USA: Prentice Hall PTR.

Heller, K., Svore, K., Keromytis, A. D., & Stolfo, S. (2003). One class support vector machines for detecting anomalous windows registry accesses. In *Workshop on Data Mining for Computer Security (DMSEC), Melbourne, FL, November 19, 2003* (pp. 2–9).

Hu, W., Liao, Y., & Vemuri, V.-R. (2003). Robust support vector machines for anomaly detection in computer security. *In proceedings of the 2003 international conference on machine learning and applications* (pp. 168–174).

Huang, C.-J., Chen, Y.-J., Chen, I.-F., & Wu, T.-H. (2009). An intelligent infotainment dissemination scheme for heterogeneous vehicular networks. *Expert Systems with Applications, 36*(10), 12472–12479.

Johnson, D. B., & Maltz, D. A. (1996). Dynamic source routing in ad hoc wireless networks. *Mobile Computing*, 153–181.

Josang, A., & Pope, S. (2012). Dempster's rule as seen by little colored balls. *Computational Intelligence, 28*(4), 453–474.

Kaplantzis, S., Shilton, A., Mani, N., & Sekercioglu, Y.-A. (2007). Detecting selective forwarding attacks in wireless sensor networks using support vector machines. In *Proceedings of the 3rd international conference on intelligent sensors, sensor networks and information* (pp. 335–340). Melbourne, Australia: IEEE.

Kargl, F., Klenk, A., Weber, M., & Schlott, S. (2004). Sensors for detection of misbehaving nodes in MANETs. In *Proceedings of workshop detection of intrusions and malware & vulnerability assessment: 46*.

Kaur, M., Kaur, S., & Singh, G. (2012). Vehicular ad hoc networks. *Journal of Global Research in Computer Science, 3*(3), 61–64.

kaushik, R., & Singhaii, J. (2011). Detection and isolation of reluctant nodes using reputation based scheme in an ad-hoc network. *International Journal of Computer Networks & Communications (IJCNC), 3*(2), 95–105.

Khan, U., Agrawal, S., & Silakari, S. (2015). Detection of malicious nodes (DMN) in vehicular ad-hoc networks. *Procedia Computer Science, 46*, 965–972.

Konar, A., Chakraborty, U.-K., & Wang, P.-P. (2005). Supervised learning on a fuzzy petri net. *Information Sciences, 172*(3-4), 397–416.

Konorski, J., & Orlikowski, R. (2009). Data-centric Dempster–Shafer theory-based selfishness thwarting via trust evaluation in MANETs and WSNs. In *Ntms* (pp. 1–5). IEEE.

Kumar, N., & Chilamkurti, N. (2014). Collaborative trust aware intelligent intrusion detection in VANETs. *Computers & Electrical Engineering, 40*(6), 1981–1996.

Laskov, P., Gehl, C., Krüger, S., & Müller, K.-R. (2006). Incremental support vector learning: analysis, implementation and applications. *Journal of Machine Learning Research, 7*, 1909–1936.

Lee, S., Pan, G., Park, J., Gerla, M., & Lu, S. (2007). Secure incentives for commercial ad dissemination in vehicular networks. In *Proceedings of the 8th acm international symposium on mobile ad hoc networking and computing* (pp. 150–159).

Lee, S., Park, J., Gerla, M., & Lu, S. (2012). Secure incentives for commercial ad dissemination in vehicular networks. *IEEE Transactions on Vehicular Technology, 61*(6), 2715–2728.

Li, F., & Wu, J. (2009). Frame: an innovative incentive scheme in vehicular networks. In *Proceedings of the 2009 ieee international conference on communications* (pp. 4638–4643).

Li, W., Anupam, J., & Tim, F. (2010). Smart: an SVM-based misbehavior detection and trust management framework for mobile ad hoc networks. *IEEE UMBC Tech report CS-TR-11-01*.

Li, W., & Joshi, A. (2009). Outlier detection in ad hoc networks using Dempster–Shafer theory. In *Mobile data management* (pp. 112–121). IEEE Computer Society.

Lian, Q., Peng, Y., Yang, M., Zhang, Z., Dai, Y., & Li, X. (2008). Robust incentives via multi-level Tit-for-Tat. *Concurrency and Computation:Practice and Experience, 20*, 167–178.

Lin, M., Sun, X., Ho, P.-H., & Shen, X. (2007). GSIS: a secure and privacy preserving protocol for vehicular communications. *IEEE Transactions on Vehicular Technology, 56*(6), 3442–3456.

Liu, J. K., Yuen, T. H., Au, M. H., & Susilo, W. (2014). Improvements on an authentication scheme for vehicular sensor networks. *Expert Systems with Applications, 41*(5), 2559–2564.

Marti, S., Giuli, T. J., Lai, K., & Baker, M. (2000). Mitigating routing misbehavior in mobile ad hoc networks. In *Proceedings of the 6th annual international conference on mobile computing and networking* (pp. 255–265).

Massimiliano, P., Alessandro, V., & Roi, V. (1997). Properties of support vector machines. In *Neural computation archive* (pp. 955–974).

Michiardi, P., & Molva, R. (2002). CORE: a collaborative reputation mechanism to enforce node cooperation in mobile ad hoc networks. In *Proceedings of the ifip tc6/tc11 sixth joint working conference on communications and multimedia security* (pp. 107–121).

Narendra, M.-M., & Savita, C. (2014). Comparative study of simulators for vehicular ad-hoc networks (VANETs). *International Journal of Emerging Technology and Advanced Engineering, 4*(4), 528–537.

Perrig, A., Canetti, R., Tygar, J. D., & Song, D. (2005). The tesla broadcast authentication protocol. *RSA CryptoBytes, 5*.

Pham, H., & Triantaphyllou, E. (2008). The impact of overfitting and overgeneralization on the classification accuracy in data mining. In *Proceedings of the soft computing for knowledge discovery and data mining* (pp. 391–431).

Scholkopf, B., & Smola, A.-J. (2001). *Learning with kernels: support vector machines, regularization, optimization, and beyond (adaptive computation and machine learning)* (1st). The MIT Press.

Schubert, J. (2011). Conflict management in dempster-shafer theory using the degree of falsity. *International Journal of Approximate Reasoning, 52*(3), 449–460.

Schlkopf, B., Sung, K.-K., Burges, C., Girosi, F., Niyogi, P., Poggio, T., & Vapnik, V. (1997). Comparing support vector machines with Gaussian kernels to radial basis function classifiers. *IEEE Transactions on Signal Processing, 45*(11), 2758–2765.

Sedjelmaci, H., & Feham, M. (2011). Novel hybrid intrusion detection system for clustered wireless sensor network. *International Journal of Network Security & Its Applications, 3*(4).

Sedjelmaci, H., & Senouci, S. M. (2015). An accurate and efficient collaborative intrusion detection framework to secure vehicular networks. *Computers & Electrical Engineering, 43*, 33–47.

Shea, C., Hassanabadi, B., & Valaee, S. (2009). Mobility-based clustering in VANETs using affinity propagation. In *Proceedings of the 28th ieee conference on global telecommunications*. In *GLOBECOM'09* (pp. 4500–4505). IEEE Press.

Shon, T., & Moon, J. (2007). A hybrid machine learning approach to network anomaly detection. *Information Sciences, 177*(18), 3799–3821.

Sung, A.-H., & Mukkamala, S. (2003). Identifying important features for intrusion detection using support vector machines and neural networks. In *Proceedings of the symposium on applications and the internet* (pp. 209–217).

Tsang, I. W., Kwok, J. T., & Cheung, P.-M. (2005). Core vector machines: fast SVM training on very large data sets. *Journal of Machine Learning Research, 6*, 363–392.

Vapnik, V.-N. (1995). *The nature of statistical learning theory*. New York, NY, USA: Springer-Verlag New York, Inc..

Wahab, O. A., Bentahar, J., Otrok, H., & Mourad, A. (2015). A survey on trust and reputation models for web services: single, composite, and communities. *Decision Support Systems, 74*, 121–134.

Wahab, O. A., Otrok, H., & Mourad, A. (2013a). A Dempster–Shafer based tit-for-tat strategy to regulate the cooperation in VANET using QoS-OLSR protocol. *Wireless Personal Communications, 75*(3), 1635–1667.

Wahab, O. A., Otrok, H., & Mourad, A. (2013b). VANET QoS-OLSR: qos-based clustering protocol for vehicular ad hoc networks. *Computer Communications, 36*(13), 1422–1435.

Wahab, O. A., Otrok, H., & Mourad, A. (2014). A cooperative watchdog model based on Dempster–Shafer for detecting misbehaving vehicles. *Computer Communications, 41*, 43–54.

Wang, S.-S., Yan, K.-Q., Wang, S.-C., & Liu, C.-W. (2011). An integrated intrusion detection system for cluster-based wireless sensor networks. *Expert Systems with Applications, 38*(12), 15234–15243.

Yi, Y., Wu, J., & Xu, W. (2011). Incremental svm based on reserved set for network intrusion detection. *Expert Systems with Applications, 38*(6), 7698–7707.

Zhang, L., Wu, Q., Solanas, A., & Domingo-Ferrer, J. (2010). A scalable robust authentication protocol for secure vehicular communications. *IEEE Transactions on Vehicular Technology, 59*(4), 1606–1617.

Zhong, S., Yang, Y., & Chen, J. (2003). Sprite: a simple, cheat-proof, credit-based system for mobile ad hoc networks. In *Proceedings of infocom 2003* (pp. 1987–1997).