Securing Connected and Autonomous Vehicles with Sliding Window-based False Information Tracing

Abinash Borah
Department of Computer Science
Oklahoma State University
Stillwater, Oklahoma, USA
aborah@okstate.edu

Anirudh Paranjothi
Department of Computer Science
Oklahoma State University
Stillwater, Oklahoma, USA
anirudh.paranjothi@okstate.edu

***Abstract*—Connected and Autonomous Vehicles (CAVs) depend on cooperation between vehicular nodes and infrastructures and offer various benefits through collaborations. The security of these cooperative wireless networks should be ensured to safeguard the network users and other on-road entities. Compromised vehicles sending false information is one such security threat for CAVs. The existing methods for detecting information falsification in vehicular networks have limitations, including overlooking collusion among malicious nodes, relying on the assumption that the majority of nodes are always honest, requiring past traffic or vehicular data, and exhibiting high computational complexity. Further, these approaches do not consider some potential problems in vehicular networks, such as message corruption, or do not evaluate the behavior of the nodes in an adequate timeframe. We present a framework, called Sliding Window-based Information Falsification Tracing (SWIFT), for securing CAVs from false information. SWIFT addresses the limitations of the existing approaches and experimental comparisons in networks with high percentages of malicious nodes show that it achieves a 24% lower data processing time and up to 46% lower false positive rate than comparative approaches.**

Keywords—connected and autonomous vehicles, clustering, false information detection, security, unsupervised learning.

I. Introduction

Connected and Autonomous Vehicles (CAVs) depend on cooperation between vehicular nodes and infrastructures, like Road Side Units (RSUs). Vehicular nodes transmit beacon messages at specific intervals that include their real-time details, including position, speed, acceleration, and direction. The communication between vehicles and infrastructures offers various benefits, such as improved road safety, traffic efficiency, efficient emergency response, and better driving experiences. However, similar to other types of networks, vehicular networks also face a variety of security threats [1, 2]. The unique characteristics of vehicular networks, such as rapid movement, constantly changing topology, and irregular connectivity, further complicate the efforts to secure them.

One of the key security challenges of vehicular networks is the potential existence of compromised nodes within the network that spread false information about fake road incidents, like accidents or traffic jams. Furthermore, groups of such nodes can collude and introduce coordinated information falsification to increase the perceived credibility of a fake incident [3]. Detecting information falsification is critical for ensuring public safety and preventing potentially dangerous reactions of vehicles to false emergency alerts.

Current solutions to trace false information in vehicular networks use a variety of techniques, such as machine learning, statistical analysis, blockchain, and trust-based systems. However, these methods come with certain disadvantages. They often overlook coordinated attacks by multiple malicious nodes, assume that most of the nodes are always trustworthy, depend on infrastructure that is not available everywhere, require inter-vehicle coordination without looking into each other's reliability, rely on historical traffic or vehicle data, or are computationally intensive. Moreover, current approaches ignore potential problems in vehicular networks, such as message corruption or sensor faults in vehicles, which may result in incorrectly identifying honest nodes to be malicious for isolated false messages stemming from data corruption or temporary faults in vehicles. Furthermore, the approaches evaluating the behavior of vehicles over an extended period do not adequately discard their past behavior. This can allow a malicious node to first establish its repute before spreading false information and going undetected due to its past reputation [4].

The RSU-based intrusion detection mechanism proposed in [5], hereafter referred to as RSUOIDM, utilizes historical data from the locality of RSUs to construct anomaly detection models. These models are subsequently employed by RSUs to assess incoming messages through anomaly scores derived from deviations relative to the established model. However, the reliance on historical data introduces initialization latency, and variations in traffic patterns between data collection and deployment times can degrade the system's accuracy. The data clustering-based false message detection technique introduced in [6], hereafter denoted as DCFMD, identifies false messages by clustering nodes based on similarities in beacon message information. Similar to RSUOIDM, DCFMD also depends on RSUs and presumes that malicious nodes constitute a minority, an assumption that may not consistently hold in real-world scenarios. The trust-based approach presented in [7], referred to as ITMS, establishes a trust management framework where nodes evaluate received messages by combining direct trust, based on their own experiences, with indirect trust obtained from neighboring nodes. However, ITMS exhibits a high false alarm rate when the proportion of malicious nodes increases and requires inter-vehicle trust dependency, which may not be reliable under collusion scenarios.

In this research, we address the limitations of existing approaches for false information detection in vehicular networks by adopting an unsupervised learning approach, data clustering, to detect colluded information falsification without

979-8-3503-9281-4/25/$31.00 ©2025 IEEE

assuming an honest majority. We further use a sliding window to trace the behavior of the nodes in the recent past, giving higher weightage to the most recent behavior of the nodes in comparison to their past behavior. Thus, we name our approach Sliding Window-based Information Falsification Tracing (SWIFT). SWIFT clusters real-time beacon message data to discover false information without depending on historical traffic information. Any node in the network can independently execute SWIFT to uncover false information transmitted by malicious nodes in its neighborhood.

The *novelty* of SWIFT comes from detecting false information, including collusions, with only real-time network statistics, with a weighted sliding window-based evaluation. This is achieved without requiring collaboration from other vehicles or roadside infrastructures, and without presuming an honest majority. The *motivations* for SWIFT are to facilitate independent discovery of false information by any node, even when malicious nodes are in the majority, considering the possibility of message corruption or temporary faults in vehicles. Experimental evaluations with simulations show that SWIFT achieves a 24% lower data processing time and up to 46% lower false positive rate than comparative methods in [5-7] at high percentages of malicious nodes.

The contributions of this research are: 1) We present an information falsification tracing method for CAVs, SWIFT, using data clustering that nodes in a network can execute independently; 2) SWIFT works with only real-time network characteristics, considers collusion among attackers, avoids the assumption of an honest majority, and uses a weighted sliding window-based evaluation; 3) We carry out extensive performance comparison of SWIFT with three existing approaches in [5-7].

The rest of the paper is presented as follows: Section II discusses recent related work; Section III describes the specifics of the SWIFT framework; Section IV presents the performance comparisons; and Section V offers the concluding remarks.

II. Related Work

We now review some recent works on false information detection in vehicular networks and discuss their limitations.

The method proposed in [3] builds a time series of vehicles' traffic parameters and applies the long short-term memory model to classify the time series to differentiate between genuine and false events. The approach presented in [8] uses the features of vehicles extracted from signal properties and uses the Kalman filter algorithm to find contextual information patterns for vehicles. An artificial neural network classifier is trained using the innovation errors from the Kalman filter to identify false information. The falsification detection method introduced in [9] utilizes an ensemble learning strategy, a random forest classifier, with parameters optimized through randomized search. While experimental results demonstrate high detection accuracy, the study does not assess the processing latency of their approach. These approaches [3, 8, 9] have the limitation of using historical traffic data to train their respective classifiers, making it impractical for real-time deployment.

The technique proposed in [10] uses the vehicles' on-board units to form a fog layer under the control of a centralized guard node to evaluate the speeds reported by vehicles with a statistical method. Another statistical method with fog computing is presented in [11], where a dynamic fog layer is deployed with the help of vehicles parked next to a road. The fog nodes continuously collect data from nearby vehicles, compute their average speeds, and perform statistical tests to recognize malicious behavior. Both these methods [10, 11] assume that the majority of the nodes in the network are honest.

A reputation system built on blockchain is presented in [12], in which vehicles verify events shared by others to assess reputations and store them in a blockchain maintained by RSUs. The study in [13] introduces a reputation management framework involving two blockchains, the first blockchain maintained by vehicles and the second one by RSUs. This method evaluates information shared based on a combination of direct and indirect trust derived from historical reputation. The research in [14] introduces a trust management system with blockchain to assess the credibility of vehicles and the data they transmit to identify false information. When vehicles notify any events to nearby RSUs, the system employs a trust model to validate the reports. The RSUs then work in coordination to update the trust scores of vehicles on the blockchain. The use of blockchains makes these approaches [12-14] computationally expensive, and they also assume an honest majority.

The proposed framework, SWIFT, addresses the limitations of the current approaches [3, 5-14]. SWIFT uses only real-time network data to detect information falsification. In addition, a node can independently execute SWIFT without requiring collaboration from other nodes or any roadside infrastructure.

III. The Proposed Framework: SWIFT

This section explains the details of SWIFT, beginning with a description of the false information model.

False information model: To establish the illusion of road congestion or on-road emergencies, malicious vehicles deliberately broadcast speeds that are lower than their real speeds in the beacon messages [10]. Additionally, a group of malicious nodes collude and transmit similar false speeds to emulate a state of movement after a traffic event [2, 12].

A. Overview of SWIFT

In the SWIFT framework, vehicular nodes analyze data in the beacon messages received from nearby nodes to identify false information. Each node keeps a list of neighboring nodes, N, which includes their IDs, speeds, and locations. The neighbor list can be evaluated continuously to detect false speed data, based on the idea that vehicles in the same range usually travel at comparable speeds due to shared traffic conditions and movement patterns influencing each other. Therefore, if certain nodes report speeds that significantly differ from the other nodes in the vicinity, they may be flagged as malicious. To do this, an evaluator node first groups the nodes in its neighbor list into clusters based on the similarities in their speed and location. Then, during the evaluation phase of SWIFT, these clusters, along with any nodes that do not fit into any cluster, are analyzed to identify falsified messages. For each node, the status of evaluation for a set of recent messages, whether genuine or false, is maintained in a separate sliding window. After receiving each beacon message from a node, the node's sliding

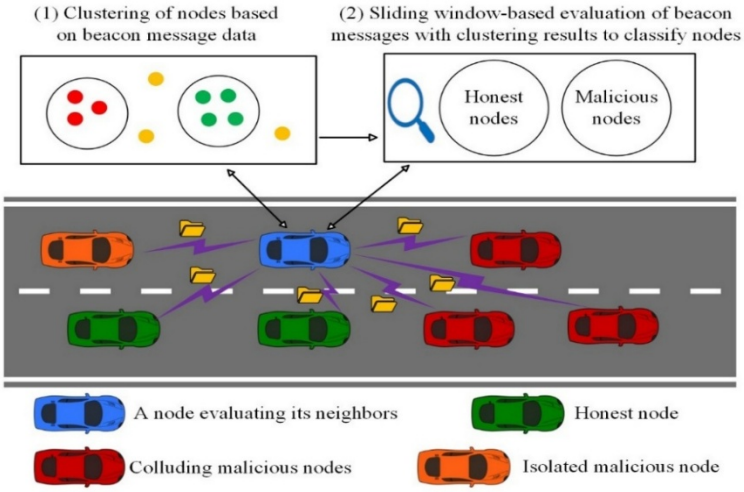


Fig. 1. The overall process of the SWIFT framework.

window is updated with the status of evaluation of the message, and a weighted anomaly score is computed from the data in the window. At any point, if the anomaly score for a node exceeds a specified threshold, the node is flagged as malicious. Figure 1 illustrates the overall process of SWIFT. The following sections provide a detailed explanation of the stages of SWIFT.

B. Clustering Phase

We employ the widely used Density-Based Spatial Clustering of Applications with Noise (DBSCAN) [15] algorithm in SWIFT due to its alignment with the specific requirements of our problem. A key advantage of DBSCAN is that it does not require prior knowledge of the number of clusters, which is suitable given that the number of clusters is not known in advance in our scenario. Additionally, DBSCAN is effective in handling noise and outliers, an essential feature for identifying isolated or non-colluding malicious nodes. The clustering process is based on three vehicle attributes from the neighbor list: speed, latitude, and longitude. Since these attributes are measured on different scales, we apply Min-Max normalization, as defined in equation (1), to ensure consistent distance computation. Through this normalization, each attribute value for all the vehicles maps to the range [0, 1], and thus equally contributes to distance measurement in DBSCAN.

$$x' = \frac{x - x_{min}}{x_{max} - x_{min}} \quad (1)$$

Here, x' is the normalized value of the attribute x and x_{min} and x_{max} respectively are the minimum and maximum values for x among all nodes in the node list.

In our framework, DBSCAN initiates clustering by selecting a random, unvisited node from the neighbor list and determining whether it has a sufficient number of neighboring nodes (*minPts* parameter of DBSCAN) within a specified radius ϵ to form a cluster. Euclidean distance is used to calculate proximity between nodes. If a node has at least *minPts* neighbors within this distance, it is designated a core point. Clusters are then formed by aggregating all nodes within a radius of ϵ from core points to the same cluster, and recursively expanding the clusters. This process continues iteratively until every node in the neighbor list has been traversed. The *minPts* value is chosen following the general guideline for DBSCAN, in equation (2).

$$minPts \geq D + 1 \quad (2)$$

Here, D is the number of attributes in the data. As the neighbor list data is three-dimensional, including speed, latitude, and longitude of vehicles, we choose $minPts = 4$. Also, considering a small value for *minPts* enables SWIFT to detect collusions in small-sized groups among malicious nodes.

During the clustering phase, nodes with similar values for location and speed are grouped into the same cluster, resulting in a set of clusters C and a set of unclustered nodes U, commonly referred to as noise points in DBSCAN. These outputs are subsequently analyzed in the evaluation phase of SWIFT, as detailed in section III.C. To cope with the dynamic changes in the movement patterns of vehicles, including speed and traffic density variations, the clustering is performed periodically, and the updated results are used in the evaluation phase.

C. Evaluation Phase

In the evaluation phase of SWIFT, an evaluator node first identifies the cluster whose members exhibit speeds that are most similar to its speed, under the assumption that such nodes are more likely to be trustworthy. For each cluster, the evaluator computes the average speed of its constituent nodes. The cluster whose average speed is closest to that of the evaluator itself is designated as the most trusted cluster, c_t, and all nodes in c_t are deemed honest. In cases where two clusters have average speeds equidistant from the evaluator's speed, the cluster with the higher average speed is selected as c_t. This is based on the false information model assumption that malicious nodes report falsely reduced speeds to imitate fake events.

The average speed of nodes in the trusted cluster c_t is used as the reference speed for assessing the remaining clusters in C and the set of unclustered nodes U. For each cluster in C excluding c_t, if the cluster's average speed is below the reference speed by more than a dynamically determined threshold T (as detailed in the following paragraph), the nodes in that cluster are classified as colluding attackers. Otherwise,

the cluster is deemed honest. Similarly, each node in the unclustered node set U is individually evaluated by comparing its speed to the reference speed from c_t, and nodes whose speeds differ from the reference by more than T are identified as isolated malicious nodes sending false messages. As clustering is performed periodically, for evaluating a beacon message in the interval between two rounds of clustering, first, it is checked whether the sender node should belong to a cluster. If so, the status for the cluster, honest or malicious, is used to classify the message. If the node does not belong to any cluster, evaluation is performed by directly comparing the speed value in the message with the reference speed.

The speed threshold T is dynamically calculated by an evaluator node based on the average speed of nodes in the trusted cluster c_t, taking into account the specific traffic conditions. For example, greater speed variability is expected when vehicles are moving at higher speeds compared to lower-speed scenarios. To capture this variability, T is calculated as the root mean square deviation of the average speeds of all the remaining clusters, as well as the individual speeds of nodes in the unclustered set U, from the average speed of nodes in c_t. Thus, if a_t is the average speed of the nodes in c_t, $a_1, a_2, a_3, \ldots$ are the average speeds for the other clusters, and $s_1, s_2, s_3, \ldots$ are the speeds of the individual unclustered nodes in U, the threshold T is computed as shown in equation (3).

$$T = \sqrt{\frac{(a_1-a_t)^2+(a_2-a_t)^2+\ldots+(s_1-a_t)^2+(s_2-a_t)^2+\ldots}{|C|+|U|}} \quad (3)$$

That is, T is computed in a manner analogous to standard deviation, with the distinction that deviations are calculated relative to the average speed a_t of the nodes in the trusted cluster, rather than the overall average speed of all nodes. Also, for each cluster identified during the clustering phase, only a single representative value, the cluster's average speed, is used in the computation, rather than individual node speeds.

It is important to note that the number of nodes within each cluster does not affect the computation of T. Regardless of cluster size, each cluster contributes only a single representative value, its average speed, to the threshold calculation. Consequently, even if a group of malicious nodes forms a large or majority cluster, their collective influence on T remains limited. This ensures that such collusion does not affect the evaluation criteria.

In the sliding window maintained for each node, the status of evaluation for a set of s (called the window size) most recent beacon messages from the respective node is stored. A value of 0 is stored in the window if a message is found to be honest, and a value of 1 is stored if the message is falsified. The window is a first-in-first-out data structure that is continuously updated after evaluating each beacon message from the node by discarding the status of the oldest message and inserting the status of the newest message. After each update to the window, a weighted anomaly score A is computed from the data in the window using a weight factor ω ($0 < \omega < 1$) using equation (4), where $W[i]$'s are elements of the window. ω is used to discard the old behavior of a node and give more weight to the recent behavior.

$$A = \sum_{i=0}^{s-1} W[i]\omega^{s-1-i} \quad (4)$$

Algorithm 1: Overall procedure of the SWIFT framework

Input: Neighbor list N (*ID*, *speed*, *latitude*, *longitude*), Distance ϵ, Min. points *minPts*, Window size s, Weight factor ω
Output: Set of malicious nodes M

$M = \{\}$
Normalize *speed*, *latitude*, *longitude* using (1)
(C, U) = DBSCAN (N, ϵ, *minPts*)
for each cluster $c_i \in C$
  Compute $average(speed)_i$
end for
Find c_t as $min\ (|average(speed)_i - own_{speed}|)$, $c_i \in C$
Compute speed threshold T using (3)
for each $c_j \in C - \{c_t\}$
  if $average(speed)_t - average(speed)_j > T$
    Mark nodes in c_j as a group of colluding nodes
    for each $ID \in c_j$
      Initialize window W with size s for node ID
    end for
  end if
end for
for each $ID \in U$
  if $average(speed)_t - speed_{ID} > T$
    Initialize window W with size s for node ID
  end if
end for
for each subsequent beacon message from node ID
  if ID belongs to an honest cluster
    continue
  else if ID belongs to a malicious cluster
    Delete the first element from the window for ID
    Insert 1 at the end of the window for ID
    Compute anomaly score A for ID
    if $A > \frac{1-\omega^s}{2(1-\omega)}$, $M = M \cup ID$
  else if $average(speed)_t - speed_{ID} > T$
    Delete the first element from the window for ID
    Insert 1 at the end of the window for ID
    Compute anomaly score A for ID
    if $A > \frac{1-\omega^s}{2(1-\omega)}$, $M = M \cup ID$
  end if
end for
Output M
end

It can be observed from equation (4) that the maximum value of A can be $\frac{1-\omega^s}{1-\omega}$. SWIFT classifies a node as malicious while evaluating its sliding window if its anomaly score exceeds half of this maximum value, indicating that at least half of the recent messages from the node should be honest, accounting for issues such as message corruption. The complete procedure of SWIFT is outlined in Algorithm 1.

IV. Performance Evaluation

We discuss the performance evaluations of SWIFT in this section. We compare the results to three techniques: RSUOIDM [5], DCFMD [6], and ITMS [7], as stated earlier in section I.

A. Simulation Setup

The evaluations are performed using the Veins framework [16] that supports the simulation of vehicular networks with SUMO and OMNET++ simulators. SUMO creates traces of vehicle movements, and OMNET++ establishes vehicular communication. For the simulations, a map of the city of Stillwater in the US state of Oklahoma is taken from OpenStreetMap with a section of highway U.S. 177 passing through the city. We run 500 vehicles that travel at 30-45 mph. To emulate road congestion, malicious vehicles broadcast falsified lower speeds in their beacon messages. The proportion of malicious vehicles is changed in the 10% to 60% range. The parameter ϵ for DBSCAN is set at 0.4 through experiments, as this value results in the optimal detection accuracy for SWIFT, as explained in subsection IV.C. Table I lists the parameter values used in the simulations.

TABLE I. PARAMETER VALUES USED IN SIMULATIONS

Sl. No.	Parameters	Values
1	Number of vehicles	500
2	Malicious node proportion	10-60%
3	Speed of travel	30-45 mph
4	Distance parameter (ϵ) for DBSCAN	0.4
5	Min points (*minPts*) for DBSCAN	4
6	Sliding window size s	10
7	Weight factor ω	0.5
8	Transmission range	500 m
9	Communication protocol	IEEE 802.11p

B. Performance Metrics

We use the following frequently used metrics to compare SWIFT with comparative methods.

Data processing time: The time taken by an evaluator node, or an RSU, to evaluate the data in beacon messages of vehicles.

Accuracy: The fraction of correctly categorized nodes out of all the nodes in the simulation setup.

Precision: The fraction of correctly classified malicious nodes among all the nodes classified as malicious.

Recall: The fraction of correctly identified malicious nodes among all malicious nodes.

F1 score: The harmonic mean of precision and recall, F1 score, reflects precision and recall in a single metric.

False positive rate (FPR): The fraction of honest nodes that are incorrectly classified as malicious.

C. Optimal Value for Distance Parameter ϵ in DBSCAN

Selecting an appropriate value for the distance parameter ϵ in the DBSCAN algorithm is crucial. If ϵ is set too low, it may result in incorrectly dividing what should be a single cluster into several smaller ones. Conversely, a high value of ϵ can cause distinct clusters to merge into one, losing important distinctions. To determine the optimal value of ϵ for SWIFT, we experiment with different values and evaluate their effect on the F1 score of SWIFT. F1 score is chosen as the evaluation metric because it balances precision and recall, expressing the capability of SWIFT to accurately detect malicious nodes while minimizing false positives.

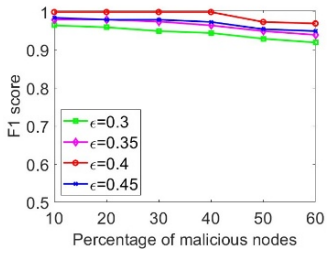


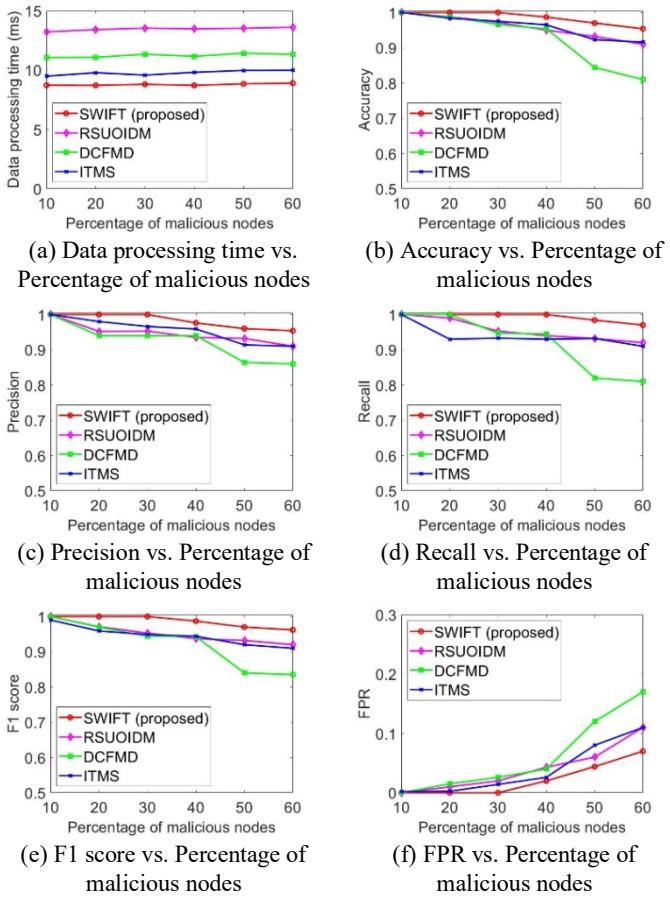
Fig. 2. Impact of varying the distance parameter ϵ in DBSCAN on the F1 score of SWIFT.

Fig. 2 shows the impact of varying the value of ϵ between 0.3 to 0.45 for our three-dimensional data on the F1 score of SWIFT. As the value of 0.4 for ϵ offers the highest F1 score, we recommend the use of this value and apply the same value in our experimental evaluations of SWIFT.

D. Results

The results of the simulations are discussed in this section in terms of the metrics discussed above.

1) Data processing time: The data processing time of SWIFT remains lower than the three comparative methods, as seen in Fig. 3(a). Compared to the other three methods, SWIFT on average achieves a 24% lower processing time. The lower processing time



(a) Data processing time vs. Percentage of malicious nodes

(b) Accuracy vs. Percentage of malicious nodes

(c) Precision vs. Percentage of malicious nodes

(d) Recall vs. Percentage of malicious nodes

(e) F1 score vs. Percentage of malicious nodes

(f) FPR vs. Percentage of malicious nodes

Fig. 3. Performance comparison results of SWIFT and comparative methods.

of SWIFT is due to the evaluation of the nodes by comparing them with clusters detected after the clustering phase. The data processing times of all four techniques are not impacted by increases in the malicious node percentage, since the amount of data that needs to be processed remains unchanged, irrespective of the malicious node percentage.

2) Accuracy: The accuracy of SWIFT remains higher than the three comparative techniques in most cases when the malicious node percentage increases, as observed in Fig. 3(b). This shows that SWIFT correctly classifies all or most nodes irrespective of the fraction of malicious nodes. The accuracy slightly reduces when the malicious node percentage rises above 30%. Among the comparative methods, the accuracy values of RSUOIDM and ITMS are comparable at a lower number of malicious nodes, whereas the accuracy of DCFMD significantly degrades due to its honest majority assumption when the malicious node percentage increases.

3) Precision: As seen in Fig. 3(c), the precision of SWIFT consistently remains higher in comparison to the other three methods [5-7]. This suggests the correctness of the false information detection approach of SWIFT. With increased percentages of malicious nodes, due to the speed variations of a fraction of honest nodes from the usual speed of honest vehicles, a small fraction of honest nodes are erroneously detected to be malicious, leading to a minor decrease in precision. While RSUOIDM and ITMS exhibit reasonable precision, DCFMD's precision significantly deteriorates at an increased number of malicious nodes.

4) Recall: The recall values for SWIFT remain perfect or near perfect until the malicious node percentage reaches above 40%, as seen from Fig. 3(d). The speed variations of malicious nodes at high malicious node percentages result in some speed values becoming close to the honest nodes, and these malicious nodes go undetected by SWIFT, leading to a slight decrease in the recall values. Nevertheless, SWIFT consistently outperforms the three comparative methods [5-7].

5) F1 score: The F1 score shows a similar pattern to precision and recall, as it reflects precision and recall collectively. As observed from Fig. 3(e), the F1 score for SWIFT is the best among the methods compared. While the F1 score of the three comparative methods remains comparable till the malicious node percentage reaches 40%, the score for DCFMD quickly degrades when the percentage of malicious nodes rises further.

6) FPR: As seen from Fig. 3(f), the FPR of SWIFT is significantly lower in comparison to the comparative methods, especially when the malicious node percentage is above 40%. Due to a very high percentage of malicious nodes, SWIFT erroneously identifies a handful of honest nodes as malicious, leading to a marginal increase in FPR. Nonetheless, SWIFT offers up to 46% lower FPR as compared to the other three methods [5-7] at high malicious node percentages. DCFMD exhibits the worst FPR when the number of malicious nodes is 50% or above due to its assumption of an honest majority.

V. Conclusions

We analyzed the limitations of the existing methods for detecting false information in CAVs, such as reliance on roadside infrastructure, dependence on past traffic data, the assumption that most nodes behave honestly, not weighing and evaluating the behavior of the nodes in an adequate timeframe, and the ignorance of potential collusion among malicious nodes in the network.

To deal with the above limitations of existing approaches, we proposed a framework called SWIFT for false information tracing with real-time data clustering and weighted sliding window-based evaluation. Nodes can independently execute SWIFT without collaboration from other nodes or roadside infrastructures. SWIFT can also detect collusion attacks and offers 24% lower data processing time on average and up to 46% lower false positive rate than the comparative approaches [5-7] in networks with high percentages of malicious vehicles. In the future, we plan to extend this framework to mitigate other security issues for CAVs.

References

[1] A. Manasrah, Q. Yaseen, H. Al-Aqrabi, and L. Liu, "Identity-based authentication in VANETs: a review," IEEE Trans. Int. Trans. Sys., vol. 26, no. 4, 2025.

[2] M. AlMarshoud, M. Sabir Kiraz, A. H. Al-Bayatti, "Security, privacy, and decentralized trust management in VANETs: a review of current research and future directions," ACM Computing Surveys, vol. 56, issue 10, 2024.

[3] Y. Yu, X. Zeng, X. Xue, and J. Ma, "LSTM-based intrusion detection system for VANETs: a time series classification approach to false message detection," IEEE Trans. Int. Trans. Sys., vol. 23, no. 12, 2022.

[4] A. Haydari, and Y. Yilmaz, "RSU-based online intrusion detection and mitigation for VANET," Sensors, vol. 22, no. 19, 2022.

[5] A. Borah, and A. Paranjothi, "Research issues in false information broadcasting rogue node detection for green VANETs," Int. Symp. on Wireless Personal Multimedia Communications (WPMC), 2023.

[6] C. Cheong, S. Li, Y. Cao, X. Zhang, and D. Liu, "False message detection in internet of vehicle through machine learning and vehicle consensus," Information Processing and Management, vol. 61, no. 6, 2024.

[7] M. E. Seno et al., "A hybrid trust management strategy for reliable cyber-physical system in intelligent transportation," IEEE Trans. Int. Trans. Sys., 2025.

[8] M. Alzahrani, M. Y. Idris, F. A. Ghaleb, and R. Budiarto, "An improved robust misbehavior detection scheme for vehicular ad hoc network," IEEE Access, vol. 10, 2022.

[9] G. O. Anyanwu, C. I. Nwakanma, J. M. Lee, and D. S. Kim, "Falsification detection system for IoV using randomized search optimization ensemble algorithm," IEEE Trans. Int. Trans. Sys., vol. 24, no. 4, 2023.

[10] A. Paranjothi, and M. Atiquzzaman, "A statistical approach for enhancing security in VANETs with efficient rogue node detection using fog computing," Digital Communication and Networks, vol. 8, no. 5, 2022.

[11] J. Hua, B. Zhang, J. Wang, X. Shao, and J. Zhu, "Rogue node detection based on a fog network utilizing parked vehicles," Applied Sciences, vol. 13, no. 2, 2023.

[12] C. P. Fernandes, C. Montez, D. D. Adriano, A. Boukerche, and M. S. Wangham, "A blockchain-based reputation system for trusted VANET nodes," Ad Hoc Networks, vol. 140, 2023.

[13] B. Hou, Y. Xin, H. Zhu, Y. Yang, and J. Yang, "VANET secure reputation evaluation & management model based on double layer blockchain," Applied Sciences, vol. 13, no. 9, 2023.

[14] W. Ahmed, D. Wu, and D. Mukathe, "Privacy preserving blockchain based authentication and trust management in VANETs," IET Networks, vol. 11, no. 3-4, 2022.

[15] M. Ester, H. Kriegel, J. Sander, and X. Xu, "A density-based algorithm for discovering clusters in large spatial databases with noise," in Proc. Knowledge Discovery Data Mining (KDD), vol. 96, no. 34, 1996.

[16] C. Sommer, R. German, and F. Dressler, "Bidirectionally coupled network and road traffic simulation for improved IVC analysis," IEEE Trans. Mobile Computing, vol. 10, no. 1, 2011.