Detecting CAM Flooding Attacks in Vehicular Networks Using Online K-means Algorithm

Harir Razzazi*, Farid Nait-Abdesselam* and Essia Hamouda†
*University of Missouri Kansas City, USA
†California State University San Bernardino, USA

***Abstract*—Vehicular Networks enable vehicles to establish communication with both other vehicles and fixed road site units for the purpose of sharing critical safety and road traffic information. These networks contribute to the development of a self-aware, efficient, secure, and partially autonomous transportation system by facilitating the exchange and collection of Cooperative Awareness Messages (CAMs). However, vehicular networks are vulnerable to security attacks, where one or more vehicles may flood neighboring vehicles with malicious CAM messages. To effectively detect such attacks, an unsupervised online learning approach, such as our proposed modified version of the online K-means algorithm, can be employed. We evaluate this method using seven datasets, achieving nearly optimal detection performance. Our approach outperforms both the traditional online K-means algorithm and three other clustering methods, yielding higher accuracy and F1-scores.**

***Index Terms*—Network security, Ad Hoc networks, Clustering, Denial-of-service attacks.**

I. Introduction

Vehicular networks, also known as Vehicular Ad Hoc Networks (VANETs), have emerged as a promising technology to enhance road safety, traffic efficiency, and overall transportation systems. Vehicular networks enable vehicles to communicate with each other and with infrastructure components, such as traffic lights and road sensors, facilitating the exchange of real-time information and enabling various applications. These applications include cooperative collision warning, intersection management, traffic congestion control, and intelligent transportation systems [1].

The foundation of vehicular networks lies in Cooperative Awareness Messages (CAMs), which vehicles exchange to share critical information about their current state, including position, speed, acceleration, and heading. CAMs play a pivotal role in enabling cooperative safety applications and facilitating efficient traffic management. By leveraging the shared information, vehicles can make informed decisions, react to potential hazards, and collaborate to improve overall road safety and traffic flow [2].

However, the widespread deployment of vehicular networks also brings forth significant security challenges and concerns. The unique characteristics of vehicular networks, such as high mobility, dynamic topology, and resource constraints, pose unique security vulnerabilities that need to be addressed. These vulnerabilities can be exploited by malicious entities, leading to potential threats and attacks that can severely impact the reliability, safety, and privacy of vehicular networks [3].

One of the primary security concerns in vehicular networks is the risk of unauthorized access and intrusion. As vehicular networks operate in an open and distributed environment, there is a possibility of unauthorized entities attempting to gain access to the network, posing as legitimate vehicles or infrastructure components. Unauthorized access can lead to various malicious activities, such as injecting false or forged CAMs, disrupting communication, manipulating traffic flow, or compromising the privacy of vehicles and their occupants [4]. Another significant security problem in vehicular networks is the threat of message tampering and data integrity violation. Since the information shared through CAMs is crucial for cooperative safety applications, ensuring the authenticity and integrity of these messages is paramount. Malicious entities may attempt to tamper with CAMs, modify their content, or inject false data, leading to inaccurate and unreliable information. Such attacks can undermine the effectiveness of safety applications and compromise the overall trustworthiness of the network [5]. Moreover, vehicular networks are susceptible to various types of denial-of-service (DoS) attacks. One of the most notable DoS attacks in vehicular networks is CAM flooding. In a CAM flooding attack, malicious entities flood the network with a large number of fake or forged CAMs, overwhelming the network's resources and causing congestion. This flood of bogus messages can disrupt the normal operation of the network, introduce communication delays, and hinder the ability of legitimate vehicles to efficiently exchange authentic safety-related information. CAM flooding attacks can result in severe consequences, such as compromised road safety, increased collision risks, and degraded network performance [6].

In this paper, we propose a modified version of the online K-means algorithm specifically designed for detecting DoS attacks on CAM messages. Our experimental results showcase the effectiveness of our method in accurately detecting CAM flooding attacks. Our approach achieved nearly optimal detection performance, outperforming other clustering methods across all evaluation metrics: (2), (3) and (4). The modified online K-means algorithm incorporates several enhancements to improve the accuracy and efficiency of attack detection. First, we introduce a feature extraction process that captures relevant characteristics from CAM messages, such as source addresses, packet sizes, and inter-arrival times. These features provide valuable insights into the traffic patterns and behavior of CAM messages, enabling the algorithm to distinguish

between legitimate and malicious traffic. Next, the algorithm utilizes a dynamic clustering mechanism that adapts to the changing network conditions and identifies clusters of CAM messages associated with potential attacks. Through the continuous updating of clusters, particularly the cluster centroids, and the reassignment of data points to these updated clusters, the algorithm effectively captures the dynamic patterns of CAM flooding attacks, thereby enhancing detection accuracy.

To assess the effectiveness of our approach, we conducted extensive experiments using seven diverse datasets spanning a range of network scenarios and attack intensities. We compared the detection results of our method with four alternative clustering methods, including the traditional online K-means algorithm, employing the same datasets. Our evaluation metrics encompass Accuracy and F-Score, which provide a comprehensive evaluation of the detection performance.

The remainder of the paper is structured as follows: Section II, provides an overview of related works. Section III focuses on the problem statement and introduces the datasets used. Section IV presented the proposed method, followed by the experiment results in Section V. Finally, Section VI concludes the paper by summarizing the findings and providing recommendations for future research.

II. Related work

Extensive research efforts have been dedicated to the detection of CAM attacks, with a particular focus on the application of machine learning (ML) models for identifying CAM flooding attacks in VANETs. In their work, Tan et al. [7] introduced a hierarchical clustering method to detect DoS attacks in vehicular networks. They defined traffic flow as a sequence of packets transmitted from source to destination within a specific time interval, extracted from Roadside Units (RSUs). By utilizing dynamic time warping as a distance measurement, they formed clusters of similar traffic flows. The authors built a model using a Python-generated dataset. Although their approach effectively reduced the authentication burden on vehicles, it incurred a notable overhead.

Singh et al. [8] presented a supervised learning system designed to detect Distributed Denial-of-Service (DDoS) attacks in a Vehicle-to-Infrastructure (V2I) scenario utilizing Software-Defined Networking (SDN) architecture. They developed a binary classifier by training various machine learning algorithms and found that the gradient boosting tree algorithm yielded the best results. However, the study lacked comprehensive information about the generated datasets and simulation scenarios, including specifics regarding the number of attacker nodes and attack rates, which limited a detailed analysis of their approach.

Yu et al. [9] introduced a machine learning-based system for detecting DDoS attacks in software-defined vehicular networks. They leveraged the OpenFlow concept to gather flow statistics and extracted features from OpenFlow tables pertaining to various transport layer protocols. Through evaluation using established datasets and training a Support Vector Machine (SVM), they achieved an accuracy rate exceeding 97%. However, it is important to note that the simulation was conducted on virtual machines, which may not fully capture the dynamic characteristics of VANETs, such as node mobility and changing topology, potentially limiting the real-world applicability of their findings.

Sharshembiev et al. [10] introduced a supervised machine learning-based system for detecting DoS attacks. They employed TensorFlow's linear classifier, specifically logistic regression, to classify network traffic as normal or malicious. The research utilized simulated data with diverse configuration scenarios. By leveraging the capabilities of TensorFlow, the proposed system aimed to accurately identify and mitigate DoS attacks in vehicular networks. Since they collect all data in a single place, it has large communication overhead.

Goncalves et al. [11] introduced an intelligent hierarchical security framework for Vehicular VANETs to address DoS and fabrication message attacks. The proposed framework utilized multiple machine learning algorithms and ensemble learning techniques, such as voting, stacking, and custom stacking, to enhance the detection capabilities. To cater to the diverse needs and characteristics of VANET entities, the authors employed different algorithms at various levels to detect attacks at multiple levels within the network. Due to the complexity of the model, it may not detect the attack in a timely manner.

In [12], the Attacked Packet Detection Algorithm to identify DoS attacks in VANETs was proposed. The algorithm was implemented in RSUs to monitor and track vehicles that send malicious packets, utilizing vehicle positions and On-Board Units (OBUs) for detection. The method demonstrated an increased overhead, delay, and reduced delivery ratio.

In [13], a method that utilizes an Enhanced Attacked Packet Detection Algorithm (EAPDA) deployed in each RSU in VANETs is proposed. The approach involves vehicles requesting traffic and other information from an RSU during a specific time slot. The RSU then compares the packet counts sent by each car and identifies malicious vehicles that are sending packets at a rate twice as high as the normal rate. These identified vehicles are subsequently excluded from further communication with the RSU. As traffic volume rises, the algorithm places excessive load on the RSU.

In [14], a packet detection algorithm for identifying nodes in a VANETs that maliciously send packets to disrupt the network's operations is proposed. The communication between nodes is facilitated through RSUs, which play a crucial role in monitoring the network. The algorithm focuses on evaluating the frequency and velocity of each node and comparing them against predetermined thresholds. Research results are rich and targeted but faces issues of costly implementation and a lack of universal applicability.

Out of the aforementioned related works, [7], [8], [9], [10] and [11] have utilized machine learning techniques; nevertheless, none of them have employed an online method. In Section V we conduct a comparative analysis between our method and these five approaches.

In Table I, we perform a comparative analysis between our approach and these five methods, with a specific focus on

TABLE I: Comparison between Modified Online K-means and other ML proposed methods to detect DoS attacks

Reference	ML Method	ML-Task	Online Update	Metrics	Memory Usage	Accuracy
Tan et al. [7]	Unsupervised	Clustering	No	DR[1]	Large	NA
Singh et al. [8]	Supervised	Binary Classification	No	TP, TN, FP, FN	Medium	NA
Yu et al. [9]	Supervised	Binary Classification	No	DR	Medium	%98
Sharshembiev et al. [10]	Supervised	Anomaly Detection	No	Pre,[2] Recall, F-1	Medium	%97
Goncalev et al. [11]	Supervised	Multi-Classification	No	TPR,[3] FPR,[4] Acc	Medium	>%92
Modified Online K-means	Unsupervised	Clustering	Yes	AUC, Pre, F-1, Acc	Low	>%98

[1] Detection Rate [2] Precision [3] True Positive Rate [4] False Positive Rate

method features, memory usage, and accuracy. Our Modified Online K-means approach excels in reducing storage requirements and incorporates the inherent advantages of online methods, while also addressing potential drawbacks associated with them.

This study introduces a novel online approach designed to facilitate prompt and precise detection of DoS attacks in VANETs. Our proposed method offers several notable advantages, including strong performance and continuous learning capabilities achieved through regular data updates. To assess the effectiveness of our approach, we perform a comprehensive comparison against a similar online method and several other clustering methods commonly employed in DoS attack detection. Through this comparison, we aim to showcase the efficacy of our proposed online method in enabling early and accurate detection of DoS attacks in VANETs.

III. Problem Statement

CAMs play a pivotal role in vehicular networks. However, they are susceptible to shortcomings, particularly flooding attacks that can disrupt network operations, as noted in [15]. To mitigate the impact of CAM flooding attacks, the implementation of robust security mechanisms and intrusion detection systems becomes imperative. These mechanisms should effectively detect and filter out fake CAM messages to ensure data authenticity. Moreover, fostering cooperation among vehicles and leveraging cryptographic techniques can further enhance network security. While these methods are effective in preventing external DoS attacks, they may not provide sufficient defense against insider attacks. Insiders can still disrupt VANETs by saturating them with encrypted messages, thereby impeding genuine communication. There are two primary types of DoS attacks: flooding, which inundates the system with messages, and crashing, which exploits vulnerabilities to induce system crashes. In Figure 1, an attacker is depicted attempting a DoS attack on nearby vehicles.

We conducted our DoS attack detection experiments based on the scenario described in [16]. The study focused on flooding attacks targeting specific entities in VANETs. They generated 21 datasets, categorized as follows: seven datasets with DoS attacks occurring within 1 to 10% of the normal data, seven datasets with attacks occurring within 10 to 20% of the normal data, and seven datasets with attacks occurring within 20 to 30% of the normal data. Each dataset has its own map size, average vehicle density, and peak vehicle density, as detailed in Tables II, III, and IV of [16]. The attack simulations

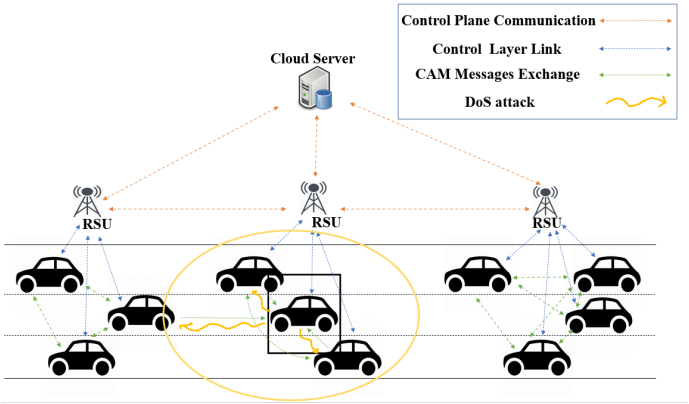

Fig. 1: DoS attack in VANET

were performed on seven different maps, and the frequency of attacker-sent messages was random. For our experiments, we selected the datasets where the DoS attacks occurred within 1 to 10% of the normal data. The data generation process followed the scenario outlined in [16].

Table V in [16] provides the parameters used for generating normal vehicle Cooperative Awareness Messages. A CAM is generated when there is a significant change in position, heading, or speed, with a minimum time interval of 100 ms between messages. However, DoS attacker nodes may deviate from these rules and send messages at a higher frequency. To detect attacks, feature extraction is performed on received CAM messages, including the time difference, position difference, speed difference, heading difference, and acceleration difference between consecutive messages from the same vehicle. These features are used as input for our intrusion detection system. All extracted features for a CAM message are treated as a data point. Through the application of our clustering approach to these data points, with storage either on a shared server or in the cloud, our objective is to achieve real-time attack detection and subsequently implement the necessary countermeasures.

IV. Proposed Method

There are various AI techniques for intrusion detection system (IDS) [17]. They can be supervised or unsupervised, offline or online. Supervised methods are normally used when samples of typical behavior are available and labeled. A certain percent of data is used for training and the test is based on the learning achieved in the training phase. In unsupervised

methods, there is no separate phase of training to learn from and the learning algorithm is left to discover the structures in the datasets. In other words, it is being trained continuously. By knowing typical attacks in advance, supervised methods can be more accurate than unsupervised ones. Though, missing instances of anomalies in the training phase may affect their accuracy. Also, in comparison with generating unlabeled data, generating labeled data is often difficult, costly, and time-consuming. The fact that the offline methods need to collect a certain amount of data and preprocess them shows that they are more time consuming and they need more computational resources, disk space, and memory.

Online methods are more adaptive to changes in the data. If a drift occurs in the test data, this method can adapt to it in real-time, while an offline method needs to wait for a new set of batch data to be processed. Regarding the advantages of online methods and unlabeled data, we used a model with such characteristics as an IDS to identify DoS attacks in VANETs.

One of the most important unsupervised learning techniques is clustering. It partitions a set of data points using some measurement of similarity (e.g., distance). The goal is to find subgroups within heterogeneous data such that each individual cluster has greater homogeneity than the whole [18]. Clustering methods are used in various applications such as anomaly detection, Ad Hoc Networks, and sensor networks. There are a variety of clustering methods, each one has its own benefits and limitations. One of those techniques is the K-means clustering model, where it is trained on unlabeled data and tries to identify similar group of data. K-means clustering can work in offline or online mode to find malicious patterns. Regarding the advantages of online K-means including popularity, simplicity, and efficiency, our contribution is the modification of traditional online K-means to improve detection of DoS attack in VANETs. In traditional online K-means, initially, as points are arriving one after another in sequence, two random points are chosen as centroids. The online algorithm will calculate Euclidean distance of each arrival point P to those centers. The closest centroid will be chosen as the center of the cluster to which this point belongs, and this centroid will be adjusted according to the following formula:

$$c_i = \frac{p + \text{clusters}[i] \times c_i}{\text{clusters}[i] + 1} \tag{1}$$

In this formula, c_i represents the centroid of cluster i. $clusters[i]$, is number of instances of cluster i. As the traditional online K-means considers all the data points it receives for tuning its centroids, we tried to use one-class SVM to find the outliers that affected the tuning of the centroids and decreased the effects of those points on adjusted centroids. One-class SVM is an unsupervised method to detect outliers by separating all the outliers from inliers using a hypersphere in the feature space and also by minimizing the volume of it, to minimize the effect of outliers in the solution [19].

In our work, the Radial Basis Function (RBF) kernel of one-class SVM assists to separate outliers from non-outliers. RBF-kernel is suitable for non-linear data and it has efficient utilization for computing memory. The one-class SVM model treats the points inside the hypersphere (inliers) to be class +1 and points outside it to be class -1.

In the algorithm, data is continuously streamed, and for each incoming point (p_j), the Euclidean distance (d) is calculated from the current center (c_i) of each cluster. The cluster of the center closest to the point is designated as the cluster for that point. The point is placed in the cluster buffer (cl_i) and the cluster center is updated using Formula 1. This process is repeated for each point until the buffer of a specific cluster contains 200 samples (num). At this juncture, we apply a one-class SVM to the filled buffer with $\gamma = 0.1$, keeping all its default hyperparameter values unchanged. The one-class SVM classifies points as outliers or inliers. Outlier points are then scaled down by a factor of 0.01 ($factor$). Subsequently, all points, including inliers, are considered, and placed into a buffer (no). The mean of these points is computed and then multiplied by a weight w_2. Meanwhile, the current cluster center is multiplied by a weight w_1. The center of the cluster is then updated to the average of these two newly computed points. For the purpose of our analysis, we selected $w_1 = 0.67$ and $w_2 = 0.33$. Following this, the cluster buffer is emptied until it is once again filled with 200 new instances for the application of the one-class SVM. The pseudocode of the algorithm is given in Algorithm 1.

V. Experiment

In our experiments, we selected K-means as our foundational method due to its simplicity, effectiveness, and widespread use. Additionally, K-means stands as an optimal solution for certain problems. However, it is essential to acknowledge the limitations of K-means, which include:

- High Sensitivity to Outliers and Data Drift: K-means is notably sensitive to the presence of outliers and variations in data over time.
- Prior Specification of Cluster Count and Initial Seed: K-means necessitates the predefined specification of the cluster count and the initial seed values.
- Prone to Local Optima: The algorithm has a tendency to converge towards local optima, leading to suboptimal solutions.

In order to address these shortcomings inherent to K-means, we propose a modified version of the method that overcomes these limitations and offers a near-optimal solution. In our particular problem context, where our aim is to detect DoS attacks rather than discerning various attack patterns, we opted for K=2 clusters, which yielded satisfactory results.

In the traditional online K-means, for each arriving point, the center of a cluster where this point belongs is updated and learning is continuously performed. The way the centroid is updated causes some abnormality. Fig 2 shows one such a case. In Fig 2, by considering a new point, marked as newly arrived outlier point, the centers will be updated, and the correct clusters shown by steady line may change to incorrect clusters as shown by dashed lines.

Algorithm 1 Modified Online K-means Algorithm

Input:
Dataset: A sequence of points p_j, $j = 0...n$,
K: Number of clusters,
w_1: Weight for scaling current cluster center,
w_2: Weight for scaling the computed mean,
num: Buffer size,
$factor$: Factor for outlier adjustment
Output:
K clusters

```
//clusters[i]: Counter of the instances of cluster i
//cl_i: A buffer for holding instances of cluster i
//no: A buffer to keep non-outliers and outliers
//so: A buffer that keeps the output of SVM application on cl_i
Set all buffers to empty.
Initialize centroids c_0 and c_1 randomly between 0 and 1.
for j = 0 to n do   // Loop over data points
    for t = 0 to K − 1 do   // Loop over clusters
        if |cl_t| ≥ num then
            so ← 1_SVM(cl_t)
            for outlier ∈ so do
                Append outlier ×factor to buffer no
            end for
            for non-outlier ∈ so do
                Append non-outlier to buffer no
            end for
            clm ← Mean(no)
            c_t ← (w_1 c_t + w_2 clm) / 2
            cl_t ← []
        end if
    end for
    Dist ← []
    for l = 0 to K − 1 do
        Dist[l] ← d(p_j, c_l)
    end for
    i ← index of Min(Dist[0], Dist[1])
    Assign p_j to cluster with centroid c_i
    c_i = (p_j + (clusters[i] × c_i)) / (clusters[i] + 1)
    Append p_j to cl_i
    clusters[i] ← clusters[i] + 1
end for
```

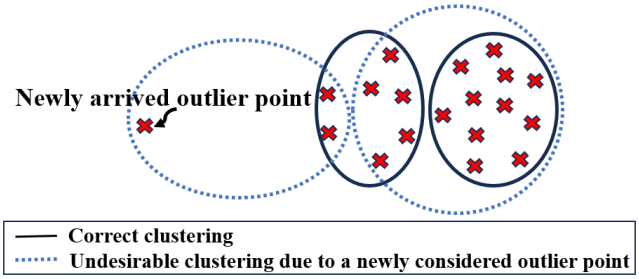

Fig. 2: Online K-means clustering sensitivity to outliers

In Table II, we also compared performance of our method in detecting DoS attacks, with Traditional Online K-means, Gaussian Mixture Model (GMM) [20], MiniBatchKmeans [21], and offline K-means on the seven datasets using precision, F1-score and accuracy given in Equations 2, 3 and 4 [22].

$$Precision = \frac{TP}{TP + FP} \tag{2}$$

$$F1 - scores = \frac{2 \times \text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \tag{3}$$

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{4}$$

In Table II, the Time and Acc (Accuracy) show the trade off between obtaining better accuracy and the usage of CPU time. We ran MiniBatchKmeans three times with different initialization values and included the best results in Table II. The results presented in Table II indicate that, in terms of accuracy (4), our method outperforms the four methods across all the considered datasets. Regarding the F1-score (3), our method outperforms all other methods for all datasets, except for a slight difference between our method and GMM in Dataset 1. In terms of precision (2), the GMM method yields results with a slightly better performance than ours in datasets 1, 2, 3, 4, and 6. Furthermore, as evident from our experiments on Dataset 'Out_7,' our method excels at clustering very large datasets with imbalanced data in a reasonable amount of time. It consistently outperforms the four other methods across all three evaluation metrics: F1-scores, precision, and accuracy.

TABLE II: Accuracy of each model using various datasets

Row	Dataset	N-Instances	N-attacks	N-Normal	Model	Time (sec)	Pre	F-1	Acc
1	Out_1	65457	42264	23193	Modified Online	00:03.94	96.96	98.43	98.90
					traditional Online	00:0066	76.57	86.71	91.68
					GMM (diag)	00:01.97	99.98	98.52	98.12
					MiniBatchKmeans	00:0012	88.12	93.67	91.28
					Offline K-means	00:0060	75.05	85.73	91.14
2	Out_2	228653	127946	100707	Modified Online	00:12.63	98.13	98.94	99.08
					traditional Online	00:02.23	73.3	84.58	88.23
					GMM (spherical)	00:13.56	99.69	97.75	97.53
					MiniBatchKmeans	00:027	99.92	87.66	90.32
					Offline K-means	00:0043	0.0012	0.002	56.00
3	Out_3	234620	178999	55621	Modified Online	00:12.82	98.40	98.88	99.47
					traditional Online	00:02.33	77.68	87.37	94.67
					GMM (spherical)	00:07.21	99.94	96.67	95.08
					MiniBatchKmeans	00:0013	93.36	96.54	94.54
					Offline K-means	00:01.015	75.91	86.23	94.25
4	Out_4	1048575	775621	272954	Modified Online	00:57.73	99.31	98.95	99.45
					traditional Online	00:09.67	78.16	87.69	94.29
					GMM (spherical)	00:40.87	99.89	98.26	97.46
					MiniBatchKmeans	00:0025	99.85	92.47	96.35
					Offline K-means	00:01.84	0.00016	0.00032	73.97
5	Out_5	230432	150539	79893	Modified Online	00:12.74	97.64	98.76	99.15
					traditional Online	00:02.28	75.46	85.97	91.46
					GMM (spherical)	00:04.59	90.98	95.20	96.51
					MiniBatchKmeans	00:0021	94.61	97.20	96.25
					Offline K-means	00:0083	0	0	65.32
6	Out_6	413231	330231	83000	Modified Online	00:23.29	98.87	99.06	99.62
					traditional Online	00:03.91	83.98	91.25	96.76
					GMM (spherical)	00:11.98	99.97	96.49	94.58
					MiniBatchKmeans	00:0018	99.90	96.43	98.61
					Offline K-means	00:0071	0.0002	0.00048	79.91
7	Out_7	1048575	931508	117067	Modified Online	00:53.49	97.66	97.40	99.41
					traditional Online	00:09.8	76.01	86.20	97.28
					GMM (spherical)	00:25.67	71.40	83.25	95.51
					MiniBatchKmeans	00:0021	96.13	98.01	96.40
					Offline K-means	00:01.86	67.18	80.22	96.30

To evaluate the performance of our method in detecting DoS attacks, in Fig 3, we compared it with traditional online K-means in terms of Area Under The Curve (AUC) on seven datasets.

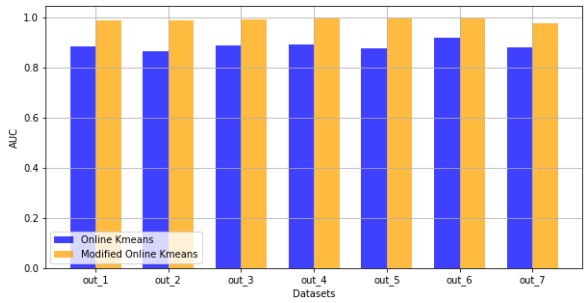

Fig. 3: AUC comparison between Online K-means and Modified Online K-means

To assess the performance of our model on both linear and nonlinear data, we utilized the Ordinary Least Squares (OLS) regression results as an evaluation metric. The R-squared value, typically used to evaluate the goodness-of-fit in linear regression models, provides insight into the strength of the linear relationship between independent and dependent variables. For datasets 1 through 7, the respective R-squared values were 0.218, 0.029, 0.035, 0.019, 0.024, 0.022, and 0.011. These values indicate the absence of a significant linear relationship between the independent and dependent variables for each dataset.

VI. Conclusion

In this paper, we presented a novel approach for detecting Denial-of-Service (DoS) attacks on CAM (Cooperative Awareness Message) messages in Vehicular Ad Hoc Networks (VANETs) by utilizing the one-class Support Vector Machine (SVM) algorithm to adjust the centroids obtained from the online K-means algorithm. Our findings indicate that this technique can significantly enhance the detection of DoS attacks on CAM messages, addressing some of the abnormalities that may arise during centroid computation in the traditional K-means algorithm.

While the weights used in our algorithm were determined empirically, the results obtained were acceptable. In future research, we plan to conduct more extensive investigations and delve deeper into this discovery with the goal of providing theoretical justifications for its effectiveness. To enhance our research, we will conduct experiments with $K \geq 2$ to identify different attack patterns and include additional performance metrics. We recognize that K-means, given its wide popularity, simplicity, and efficiency, merits further investigation and exploration in the realm of DoS attack detection.

References

[1] Ejaz Ahmed and Hamid Gharavi. Cooperative vehicular networking: A survey. *IEEE Transactions on Intelligent Transportation Systems*, 19(3):996–1014, 2018.

[2] Mate Boban and Pedro M. d'Orey. Exploring the practical limits of cooperative awareness in vehicular communications. *IEEE Transactions on Vehicular Technology*, 65(6):3904–3916, 2016.

[3] Zhaojun Lu, Gang Qu, and Zhenglin Liu. A survey on recent advances in vehicular network security, trust, and privacy. *IEEE Transactions on Intelligent Transportation Systems*, 20(2):760–776, 2019.

[4] Mohammed Saeed Al-kahtani. Survey on security attacks in vehicular ad hoc networks (vanets). In *2012 6th International Conference on Signal Processing and Communication Systems*, pages 1–9, 2012.

[5] Fadi Al-Turjman and Joel Poncha Lemayian. Intelligence, security, and vehicular sensor networks in internet of things (iot)-enabled smart-cities: An overview. *Computers Electrical Engineering*, 87:106776, 2020.

[6] Jianbing Ni, Kuan Zhang, Xiaodong Lin, and Xuemin Shen. Securing fog computing for internet of things applications: Challenges and solutions. *IEEE Communications Surveys Tutorials*, 20(1):601–628, 2018.

[7] Haowen Tan, Ziyuan Gui, and Ilyong Chung. A secure and efficient certificateless authentication scheme with unsupervised anomaly detection in vanets. *IEEE Access*, 6:74260–74276, 2018.

[8] Pranav Kumar Singh, Suraj Kumar Jha, Sunit Kumar Nandi, and Sukumar Nandi. Ml-based approach to detect ddos attack in v2i communication under sdn architecture. In *TENCON 2018-2018 IEEE Region 10 Conference*, pages 0144–0149. IEEE, 2018.

[9] Yao Yu, Lei Guo, Ye Liu, Jian Zheng, and YUE Zong. An efficient sdn-based ddos attack detection and rapid response platform in vehicular networks. *IEEE access*, 6:44570–44579, 2018.

[10] Kumar Sharshembiev, Seong-Moo Yoo, and Elbasher Elmahdi. Protocol misbehavior detection framework using machine learning classification in vehicular ad hoc networks. *Wireless Networks*, 27(3):2103–2118, 2021.

[11] Fábio Gonçalves, Joaquim Macedo, and Alexandre Santos. An intelligent hierarchical security framework for vanets. *Information*, 12(11):455, 2021.

[12] S. RoselinMary, M. Maheshwari, and M. Thamaraiselvan. Early detection of dos attacks in vanet using attacked packet detection algorithm (apda). In *2013 International Conference on Information Communication and Embedded Systems (ICICES)*, pages 237–240, 2013.

[13] Amarpreet Singh and Priya Sharma. A novel mechanism for detecting dos attack in vanet using enhanced attacked packet detection algorithm (eapda). In *2015 2nd International Conference on Recent Advances in Engineering Computational Sciences (RAECS)*, pages 1–5, 2015.

[14] Sushil Kumar and Kulwinder Singh Mann. Prevention of dos attacks by detection of multiple malicious nodes in vanets. In *2019 International Conference on Automation, Computational and Technology Management (ICACTM)*, pages 89–94, 2019.

[15] Sherali Zeadally, Ray Hunt, Yuh-Shyan Chen, Angela Irwin, and Aamir Hassan. Vehicular ad hoc networks (vanets): status, results, and challenges. *Telecommunication Systems*, 50:217–241, 2012.

[16] Fábio Gonçalves, Bruno Ribeiro, Oscar Gama, João Santos, António Costa, Bruno Dias, Maria João Nicolau, Joaquim Macedo, and Alexandre Santos. Synthesizing datasets with security threats for vehicular ad-hoc networks. In *GLOBECOM 2020-2020 IEEE Global Communications Conference*, pages 1–6. IEEE, 2020.

[17] Dilara Gumusbas and Tulay Yildirim. Ai for cybersecurity: Ml-based techniques for intrusion detection systems. In *Advances in Machine Learning/Deep Learning-based Technologies*, pages 117–140. Springer, 2022.

[18] Christoph F Eick, Nidal Zeidat, and Zhenghong Zhao. Supervised clustering-algorithms and benefits. In *16Th IEEE international conference on tools with artificial intelligence*, pages 774–776. IEEE, 2004.

[19] David MJ Tax and Robert PW Duin. Support vector data description. *Machine learning*, 54(1):45–66, 2004.

[20] Geoffrey J McLachlan and Kaye E Basford. *Mixture models: Inference and applications to clustering*, volume 38. M. Dekker New York, 1988.

[21] David Sculley. Web-scale k-means clustering. In *Proceedings of the 19th international conference on World wide web*, pages 1177–1178, 2010.

[22] Marina Sokolova and Guy Lapalme. A systematic analysis of performance measures for classification tasks. *Information processing & management*, 45(4):427–437, 2009.