Shadows Don’t Lie: n-sequence Trajectory Inspection for Misbehaviour Detection and Classification in VANETs

Anhtuan Le
Warwick Manufacturing Group
University of Warwick
Coventry, UK
a.le.1@warwick.ac.uk

Carsten Maple
Warwick Manufacturing Group
University of Warwick
Coventry, UK
cm@warwick.ac.uk

***Abstract*—This paper presents a machine learning approach to detect and classify misbehaviour in Vehicular Ad-hoc Networks. We describe three novel features obtained from analysis of n consecutive locations of a vehicle to form a judgement about its behaviour. These features are used in two machine learning algorithms (K-Nearest Neighbour and Support Vector Machine) for detecting attacks in the VeReMi dataset. We show that the overall precision rates can be as high as 99.7%, whilst the recall rates are consistently higher than 99%. The features we propose also help to reduce the overall confusion rate to less than 4.7% when classifying different types of attacks. We also show that our models can be used for effective classification after as few as 3 observations, suggesting the potential for application of the method in near real-time situations thereby improving safety and security.**

Keywords—location spoofing, trajectory inspection, VeReMi, machine learning, misbehaviour detection

I. Introduction

A. *VANET Security*

Vehicular Ad-hoc Networks (VANET) are systems that are being employed to enhance road safety and transport efficiency through the exchange of information between vehicles (V2V) and infrastructure (V2I). For example, a vehicle can send information (e.g. current location, speed, or acceleration) through Basic Safety Messages (BSMs) exchanged with other vehicles thereby improving the overall sensor’s line-of-sight and situational awareness [1]. However, such vehicular networks are susceptible to a number of security and privacy concerns [2]. One of the most critical issues is that attackers can inject falsified information into the communication stream to manipulate others’ driving decisions, leading to consequences such as false warnings, fake traffic reports, sudden braking or even accidents. Verifying the correctness of the information exchanged remains an open challenge [1].

B. *Misbehaviour Detection and Classification*

Misbehaviour attacks need to be detected to eliminate harmful sources of information. Moreover, attackers can employ numerous strategies to spoof data with the aim of achieving different objectives. Therefore, in addition to detecting attacks, classifying the attackers is also a concern if we are to understand their behaviours and improve future decision making.

The data-centric approach is one of the main techniques employed to tackle spoofing attacks, and involves collecting and analysing performance data using plausibility and consistency metrics. Some researchers have tried to predict future locations through models such as vehicle dynamics for verification, whereas others have focussed on finding malicious patterns in the data. While a number of features have been proposed for detection and classification purposes [1, 3, 4], their performance is still poor in many situations.

C. *Contributions and Paper Structure*

In this paper, we investigate the feasibility of using historical trajectory to detect and classify misbehaviour attacks. We develop machine learning models (K-Nearest Neighbour and Support Vector Machine, KNN and SVM, respectively) based on three features which can be extracted from n consecutive location observations in the trajectory data: Movement Plausibility Check, Minimum Trajectory Distance, and Minimum Translation Trajectory Distance. We evaluate the performance of our models in detecting and classifying attacks in the VeReMi dataset [5], a labelled dataset built in VEINS [6], which offers five different types of location spoofing attacks (see Table I). We compare our solution to a similar machine learning approach presented in [1]. We also vary the length of the sequences inspected to study its impacts on performance.

Our contributions are three-fold:

- We introduce three novel features to use in machine learning models, which can significantly improve performance in misbehaviour detection and classification.
- Our approach reduces the dependency on velocity data to make judgements, therefore avoiding the chance that attackers can craft useful velocity data to circumvent detection.
- We study the impact of the length of observations on detection capability. This study shows that our models can detect and classify misbehaviour attacks effectively after as few as three consecutive observations, which makes them applicable for near real-time detection.

The rest of the paper is organised as follows. Section II reviews the related work in VANET misbehaviour detection. Section III details our proposed approach. Section IV describes the experimental setup while Section V evaluates the performance. Finally, Section VI concludes the paper and presents suggestions for future work.

II. Related Work

When V2V data are used to support driving decisions, detecting falsified information being broadcasting becomes more important. Most existing detection approaches require the presence of a central authority to make judgements based on monitoring and analysing messages broadcast from all vehicles in the VANET [1].

A comprehensive survey on misbehaviour detection can be found in [4], in which the authors categorised the detection techniques into two main themes: *node-centric* and *data-centric*. The former technique focusses on verifying the interactions between nodes through information such as packet frequencies or message formats. Alternatively, the latter approach investigates the content of exchanged messages to determine validity.

This paper uses a *data-centric* approach. We focus on specifying the key plausibility features of the location information to detect and classify the senders of malicious data. A number of techniques have been developed to check the plausibility of location data. Based on previous information, some researchers have tried to estimate plausible locations and compare them with the reported locations. Margins are used for detection purposes. For example, authors in [3] have developed simple plausibility checks for fast and efficient processing. However, existing approaches have shown limited performance, while also requiring a majority of the network to be legitimate. Other researchers have tried to improve the accuracy of detection by applying more sophisticated models for estimation. For instance, researchers in [7] used a Kalman Filter while authors in [8] used the vehicle dynamics model. However, these models require all the previous data to be accurate for an accurate estimation, which is not the case when attackers inject false data consistently into the communications. Recently, plausibility checks have been used as features integrated into machine learning models (such as KNN and SVM) to detect forged locations [1]. However, these models require full journey information of the suspected vehicle before making a final decision. Moreover, some of the features rely on the velocity information broadcast from suspect vehicles, which may be crafted to disrupt detection mechanisms.

When legitimate vehicles operate in transportation, their logged trajectories are reliable reference sources to predict moving behaviour. For example, in [9], trajectory data was used to predict upcoming journeys for safety purposes. However, to the best of our knowledge, there has been little work which uses trajectory data to detect and classify location spoofing attacks. In this work, we use trajectory data as relaxed ground truths to differentiate the falsification misbehaviour of attackers rather than to accurately predict next locations.

III. Proposed Approach

In this section, we will first present definitions of basic concepts to learn from the trajectory data. We then introduce the features we have designed to detect and classify the attacks.

A. General Definition

Definition 1 *State.* The state s of a vehicle is a set of information to exchange with others. In this paper, s contains location, velocity, and time information $s = \{x, y, vx, vy, t\}$, where (x, y) is the latitude and longitude position, (vx, vy) indicates the latitude and longitude velocity, and t is the time that this information was logged.

Definition 2 *Trajectory.* The trajectory m is a set of consecutive sequences of vehicle state data:

$m = \{s_1, s_2, \dots, s_k\} = \{(x_1, y_1, vx_1, vy_1, t_1), \dots, (x_k, y_k, vx_k, vy_k, t_k)\}$, where the states are reported continuously and sorted by ascending time, which means $\forall i \in \{1, \dots, k-1\}$: $0 < t_{i+1} - t_i \le \varepsilon_t$, where ε_t is the maximum time allowed between consecutive sampling.

Definition 3 *Legitimate trajectory history.* We define the legitimate trajectory history as the set of unique trajectories recorded by all legitimate vehicles in transportation and denoted as $L = \{L_1, \dots, L_l\}$.

Definition 4 *n-sequence inspection trajectory.* Given a suspected trajectory m with k states, $k > n$, an *n-sequence* inspection trajectory m_i is a subset of n consecutive states extracted from the k states:

$$m_i = \{(x_i, y_i, vx_i, vy_i, t_i), (x_{i+1}, y_{i+1}, vx_{i+1}, vy_{i+1}, t_{i+1}), \dots, (x_{i+n}, y_{i+n}, vx_{i+n}, vy_{i+n}, t_{i+n})\} \mid i + n \le k$$

Definition 5 *Distance between two trajectories.* Given two equal *k-length* trajectories m_u and m_v, the distance $d(m_u, m_v)$ between them is a measure of how close the two trajectories are. We only consider the positions of the trajectories rather than other factors such as velocity and time. The distance will be calculated as:

$$d(m_u, m_v) = \frac{1}{k}\sum_{i=1}^{k}\sqrt{(x_i^{m_u} - x_i^{m_v})^2 + (y_i^{m_u} - y_i^{m_v})^2}$$

where $(x_i^{m_u}, y_i^{m_u})$ and $(x_i^{m_v}, y_i^{m_v})$ is the i^{th} location states of trajectory m_u and m_v respectively.

B. Feature Selection

We design the three following features to detect and classify the misbehaviour attacks: Movement Plausibility Check (MPC), Minimum Distance to Trajectories (MDT) and Minimum Translation Distance to Trajectories ($MTDT$). The MPC aims to record misbehaviours when the vehicle attackers are moving ($vx \ne 0$ or $vy \ne 0$), but reported locations are unchanged. The MDT is used to investigate whether the movement pattern of the suspected vehicles is similar to any of the logged trajectory patterns. Finally, the $MTDT$ will check whether the movement pattern of the suspected vehicles is similar to any of the logged trajectory translation.

Feature 1: MPC. Given an observed trajectory $m = \{s_1, s_2, \dots, s_n\}$ and a pre-selected constant K, for every two consecutive observation $\{s_{i-1}, s_i\}$ $(i = \overline{2,n})$, we verify the movement attack via k_i as:

$$k_i = \begin{cases} K \ if \ s_{i-1}(v) \ne 0 \wedge s_{i-1}(x) = s_i(x) \wedge s_{i-1}(y) = s_i(y) \\ 0 \quad otherwise \end{cases}$$

After calculating all k, the MPC of m will be calculated as:

$$MPC(m) = \sum_{i=2}^{n} \frac{k_i}{n-1}$$

Feature 2: MDT. Given an $n - sequence$ trajectory m and the set of legitimate trajectories L, MDT of m will be calculated as follows:

$$MDT(m, L) = \min\{d_{min}(m, L_i)\ \forall L_i \in L\}$$

where $d_{\min}(m, L_i) = \min\{d(m, L_i^n)\}\ \forall L_i^n$ as a subset of n consecutive sequences in L_i.

Feature 3: MTDT. Given an observed trajectory m with n sequences $m = \{(x_1, y_1), (x_2, y_2) \ldots, (x_n, y_n)\}$ and the set of legitimate trajectories L, the $MTDT$ between m and L_i is calculated in three steps as follows:

1) Calculate the centroid point of m: $\overline{m} = (\overline{m_x},\ \overline{m_y})$ as:

$$\overline{m_x} = \frac{1}{n}\sum_{k=1}^{n} x_k \ and\ \overline{m_y} = \frac{1}{n}\sum_{k=1}^{n} y_k$$

2) Let L_i as one of the legitimate trajectories in L, assume that L_i has q consecutive location data sequences: $L_i = \{(x_1^{L_i}, y_1^{L_i}), (x_2^{L_i}, y_2^{L_i}), \ldots, (x_q^{L_i}, y_q^{L_i})\}$. We calculate the minimum translation distance between m and L_i as follows. For every L_i^j as a subset of L_i with n consecutive observations $j = \overline{1, q-n+1}$ $L_i^j = \{(x_j^{L_i}, y_j^{L_i}), (x_{j+1}^{L_i}, y_{j+1}^{L_i}), \ldots, (x_{j+n-1}^{L_i}, y_{j+n-1}^{L_i})\}$.

2.1) Calculate the centroid point of L_i^j: $\overline{L_\iota^J} = (\overline{L_{\iota x}^J}, \overline{L_{\iota y}^J})$ as:

$$\overline{L_{\iota x}^J} = \frac{1}{n}\sum_{k=j}^{j+n-1} x_k^{L_j} \ and\ \overline{L_{\iota y}^J} = \frac{1}{n}\sum_{k=j}^{j+n-1} y_k^{L_j}$$

2.2) Calculate the translation vector as:

$$\bar{t} = (\bar{t_x}, \bar{t_y}) = (\overline{m_x} - \overline{L_{\iota x}^J}, \overline{m_y} - \overline{L_{\iota y}^J})$$

2.3) Calculate m_t (m after translation by $\bar{t}$) as:

$$m_t = \{(x_1 - \bar{t_x}, y_1 - \bar{t_y}), (x_2 - \bar{t_x}, y_2 - \bar{t_y}), \ldots, (x_n - \bar{t_x}, y_n - \bar{t_y})\}$$

2.4) The translation distance $\bar{d}$ between m and L_i^j will be the distance between m_t and L_i^j: $\bar{d}(m, L_i^j) = d(m_t, L_i^j)$.

2.5) After calculating all $\bar{d}(m,\ L_i^j)$, the minimum translation distance between m and L_i will be: $\bar{d}_{min}(m, L_i) = \min\{\bar{d}(m, L_i^j)\}\ \forall L_i^j \in L_i$ as a subset of n consecutive observations in L_i.

3) The minimum distance between m and L will be calculated as:

$$MTDT(m, L) = \min\{\bar{d}_{min}(m, L_i)\}\ \forall L_i \in L.$$

Fig. 1 illustrates the translation vector and the translation distance of two $5 - sequence$ trajectories.

IV. Experimental Setup

The VeReMi dataset has recently been published and is built specifically for testing location spoofing attacks in V2X scenarios [5]. The advantages of using the VeReMi dataset are (i) it presents different types of attackers; (ii) it simulates varied density conditions of vehicle network; (iii) it is consistent with its BSM's broadcast rate; and (iv) the scenarios are conveniently labelled for using a machine learning approach. This section will describe VeReMi and present the applications of our three designed features in machine learning for detecting and classifying attacks in this dataset.

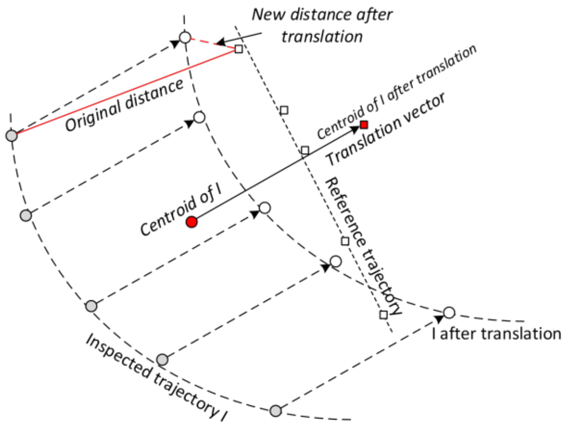

Fig. 1. Illustration of calculating MTDT

TABLE I. VeReMi Attack Types Description [1]

Type		Details	Parameters

T1	Constant	Attacker transmits a fixed location	x = 5560
			y = 5820

T2	Constant Offset	Attacker transmits a fixed, offset added to the real position	Δx = 250
			Δy = -150

T4	Random	Attacker sends a random position inside the simulation area	Uniformly random in playground
T8	Random Offset	Attacker sends a random position in a rectangle around the vehicle	Δx, Δy are uniformly random in [-300, 300]
T16	Eventual Stop	Attacker behaves normally for some time and then attacks by transmitting the same position repeatedly	Stop probability increase by 0.025 each position update

A. Attacker Model

Attackers in VeReMi are assumed to be local, insider and active, and able to operate in both uniform and non-uniform areas [1]. A local attacker can only broadcast falsified information within their defined communication range, hence the damage created should be equally local. Attackers are also assumed to be insiders so that they can communicate with other vehicles regardless of the cryptographic defence used in communication. Attackers can be active, which means they can inject falsified information in the network the way they want. Finally, *uniform areas* (e.g. highway regions) refer to the areas where the travelling information of the attackers can be consistently transmitted to other vehicles, which is the opposite of non-uniform areas (e.g. rural areas).

B. Scenario and Attack Types

The VeReMi dataset comprises of 5 position forging attacks, 3 vehicle densities (low, medium and high), 3 attacker densities (10, 20 and 30 per cent) and each parameter set was repeated 5 times for randomization. The dataset contains the message logs of the attacking and benign vehicles including reception time stamp, claimed transmission time, claimed sender, unique message ID, GPS position (x, y, z), RSSI value, position noise and speed noise vector, for each receiving vehicle in every scenario. The messages are broadcasted every second, so we chose $\varepsilon_t = 1$ for trajectory extraction (see Definition 2). Along with that, a ground truth file is also maintained which records the true values of the BSM attributes of both attacker and benign vehicles. The attacker type attribute in the Ground Truth file keeps the label of the attack ID as described in Tab. I.

C. Machine Learning Designs

Given an $n - sequence$ trajectory for inspection, it can be legitimate (referred as $T0$) or one of five types of attacks (referred as $T1, T2, T4, T8, T16$). The purposes of our designed features in detecting and classifying the attacks are as follows.

MPC aims at classifying $T1$ and $T16$ from other attacks. We have $MPC(T0, T2, T4, T8) = 0$ because $k_i = 0\ \forall i$. On the other hand, $T1$ report constant location all the time, therefore $k_i = K\ \forall i$, so $MPC(T1) = K$. Meanwhile, for $T16$, there should be at least one i so that $k_i = 0$ to indicate the time when attackers transform from normal state to attack state; while there should be at least one j that $k_j = K$ to reflect the attack. Therefore, we have $\frac{K}{n-1} \le MPC(T16) \le \frac{K(n-2)}{n-1}$ (1). Our experiments show that choosing a small K may lower the performance of KNN and SVM as the classification distances will become too small. In this work, we select $K = 1000$.

When $T1$ and $T16$ are detected, MDT will help to differentiate $T0$ from the remaining types $T2, T4, T8$ because $MTD(T0)$ values should be close to 0 as normal trajectories should find similar moving patterns in the legitimate set. On the other hand, $MDT(T2, T4, T8)$ values should be large due to the small chance of having similar movement patterns. Moreover, MDT can help to classify $T4$ from others because the $MDT(T4)$ values are much larger compared with MDT of other types.

Our experiments show that $MTD(T2)$ and $MTD(T8)$ values are close, therefore MTD alone cannot help to differentiate between $T2$ and $T8$. Therefore, we design $MTDT$ for further classifying these two types. As $T2$ is translated from a legitimate trajectory, $MTDT(T2)$ values should be close to 0. Meanwhile, $MTDT(T8)$ values will remain large due to the small chance of having movement pattern which is similar to a translated legitimate trajectory.

With the three features, we implemented similar machine learning algorithms (i.e. KNN and SVM) with [1] for comparison. In details, the KNN was done using the MATLAB built-in function $fitcknn$ with Euclidean distance to compute the nearest neighbour. The number of nearest neighbours K was tuned from 1-100, while K with the highest correct-classification-rate (CCR) was chosen as the best K for prediction model. For SVM, instead of using the binary classification $fitcsvm$ function as presented in [1], we use the $fitcecoc$ function which allow to classify for more than two types of attacks. We also use $fitcecoc$ in binary classification as our experiments show that it always performs better than the $fitcsvm$.

D. Parsed Dataset and Feature Extractions

The VeReMi dataset provides results of 225 simulations, 45 for each type of attacks. We selected 80% of the data for training while the remaining 20% are used for testing. For each type of attack, 36 sets of simulation logged data were chosen randomly to include in the training data, while the remaining 9 logged results are included in the testing data. We extracted all the unique legitimate trajectories from the 80% training dataset into a database. We assume that this database will be maintained by a central authority, where the vehicles can send their suspected observations to for querying and getting the judgement. We also extract the three features for every $n - sequence$ trajectory in the training dataset to build the machine learning models. Notes that to calculate the MTD and $MTDT$ of a trajectory, this trajectory will be excluded from the legitimate database to avoid any duplication. Finally, for testing, for every $n - sequence$ trajectory in the 20% testing dataset, we calculate its three features based on the extracted legitimate database to query with the detection and classification models. The behaviour labels of the trajectory will be used to verify the accuracy performance of the models.

E. Inspection Length Consideration

To apply our approach, we need all the three features to be valid for all inspected trajectories. It is obvious that $MPC(T16)$ is invalid when $n = 1$ or $n = 2$ according to (1). Therefore, the minimum value of n to be considered in this paper is 3. As n will influence all the values of the three features significantly, we vary n from 3 to 10 to study its impacts. Notes that when judging the behaviours of a suspected vehicle, if the central authority can gather observations from all other vehicles, n can be very large as desired. However, if the observations come from a single source, n is not likely to be too large as the suspected vehicle may move out from the communication coverage of the source, which break the continuity of the observations.

F. Evaluation Metrics

The detection performance of the approach can be evaluated through the precision and recall. Firstly, each prediction result will be counted in one of the four following categories: *True Positive (TP)*: detect an attacker as an attacker; *False Positive (FP)*: detect a legitimate observation as an attacker; *False Negative (FN)*: detect an attacker as a legitimate vehicle; and *True Negative (TN)*: detect a legitimate observation as legitimate. The *precision* is calculated as $\frac{TP}{TP+FP}$, while the *recall* is calculated as $\frac{TP}{TP+FN}$. We also formulate a confusion matrix to see the performance of the classification.

V. Results and Discussions

We evaluate our approach in a similar way to the evaluations in [1]. Firstly, we evaluate the detection for each type of attacks by combining only the corresponding attack data and the normal data to use in training and testing. For example, to evaluate the detection of $T2$, we keep only $T2$ and normal data in the training and testing dataset while excluding all other attack types. Secondly, we evaluate the overall performance in differentiating between attacks and non-attacks by generalise the attack type, which means that all the attack label 1, 2, 4, 8, 16 will be relabelled as 1. Lastly, we evaluate the classification performance for the original labelled attack dataset in two cases: with and without the normal behaviours. Notes that authors in [1] avoided the evaluation of classifications when normal behaviours were included due to unbalance dataset which led to poor performance.

A. Detection per Attack Type

Fig. 2 illustrates the detection capability of different types of attacks when applying KNN and SVM, based on [1] and

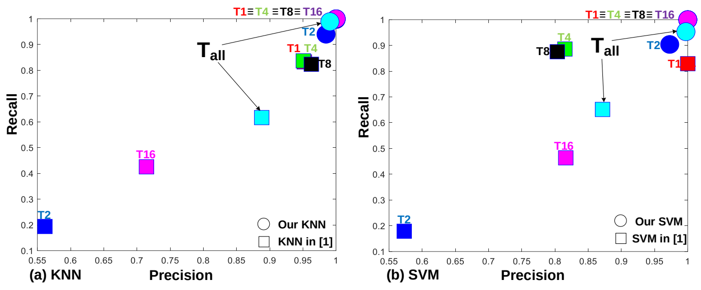

Fig. 2. Detection capability for different attack types

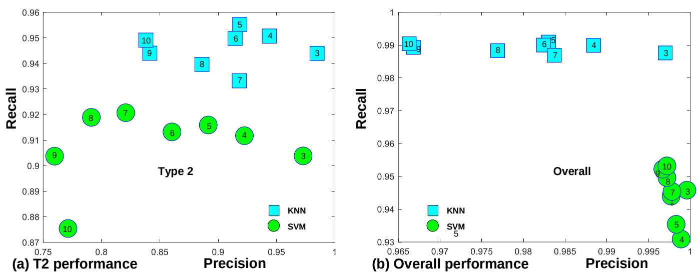

Fig. 3. T2 & overall detection capability in different n-sequence

our approach. As can be seen in the figure, our approach achieved near 100% in precision and recall for $T1, T4, T8, T16$ when applying either KNN or SVM. The MPC feature will always be 1000 if a trajectory is $T1$, and always be 0 if it is normal. For $T16$, MPC have a minimum value of 111 and maximum value of 888, both when inspecting with 10 sequences. Therefore, MPC can be considered as a plausibility check for both $T1$ and $T16$ to achieve accurate detections. On the other hand, $T4$ and $T8$ trajectories will have much larger MTD and $MTDT$ values than the normal trajectories (most of $MTD(T4)$ and $MTDT$(T4) values are larger than 1000; most of $MTD(T8)$ and $MTDT$(T8) values are larger than 100 and much smaller than 1000; while most of other $MTD(T0)$ and $MTDT(T0)$ are smaller than 20), therefore it is not difficult to detect these attacks.

For $T2$, we achieved much better precisions (the best rate is 98.5% for KNN with 3-sequence) and recalls (94% for KNN with 3-sequence) compared to [1]. [1] has low performance because it does not have any feature that can separate $T2$ from normal performance. Meanwhile, in our approach, the MTD is specifically designed to separate $T2$ from normal performance. Fig. 3a shows $T2$ detection capability with different n sequence when applying KNN and SVM. It can be seen that in general when n increase, the precision rates decrease significantly while the recall rates increase slightly. Notes that unlike other verification approaches, we do not try to predict the accurate locations of the trajectories. Instead of that, we try to find the minimum deviation of the reported trajectories compared with the normal patterns. A small n is expected to create small accumulated deviations, hence lead to smaller $MTD(T0)$ margin (upper bound values are at about 50 when $n \leq 5$ and about more than 150 when $n > 5$). On the other hand, statistics of $MTD(T2)$ are not affected much by n, because the deviations are always high. Therefore, when n is small, the models may create better separations between normal and attack dataset, which lead to higher precision. However, the smaller margin will make the model more sensitive, so the $MTD(T0)$ outliers will likely be detected as attacks, which

TABLE II. CONFUSION MATRIX OF BEST CCR KNN (N=6, K=97) (%)

		Ground Truth					
		T0	*T1*	*T2*	*T4*	*T8*	*T16*
Predicted	*T0*	98.25	0	6.39	0	0	0
	T1	0	100	0	0	0	0
	T2	1.75	0	93.61	0	0.04	0
	T4	0	0	0	100	0	0
	T8	0	0	0	0	99.96	0
	T16	0	0	0	0	0	100

TABLE III. COMPARISONS OF CONFUSION RATES (%) BETWEEN T2 AND T8 (SVM - FITCECOC)

	n = 3		n = 4		n = 5		n ≥ 6	
	T2	T8	T2	T8	T2	T8	T2	T8
T2	99.83	0.36	99.9	0.3	99.94	0.01	1	0
T8	0.17	99.64	0.1	99.97	0.06	99.99	0	1

leads to lower recall rates. When n is large, deviations will be accumulated, $MTD(T0)$ will be increased so the models will have to increase the margin, which makes more $T2$ to be detected as normal and decrease the precisions. Nevertheless, the $MTD(T0)$ outliers will less likely be detected as attacks, which make the recall rates increase slightly.

B. *Overall Detection Performance*

For evaluations in detecting attacks in general (all attacks are re-labelled as 1), Fig. 2 shows the comparisons between our approach and [1], while Fig. 3b presents the comparisons between using KNN and SVM with different length of inspected trajectories in our approach. As can be seen from the figures, our approach shows significant better performances than [1], while there are always trade-offs between the precision and recall rates when choosing different inspection length. With KNN, our approach got a precision rate of 99.7% ($n = 3$), which decreases gradually to 96.84 ($n = 10$), whilst the recall rates are around 99%. When using SVM *fitcecoc*, the precision rates are always larger than 99.7%, however, the recall rates are ranging from 93.2% ($n = 4$) to 95.2% ($n = 10$). On the other hand, when using the SVM *fitcsvm*, the precision rate is highest at 98.7% ($n = 3$), which decrease gradually to 88.77% ($n = 10$), while the recall rates are around 99%. As the model show very good performance when detecting $T1, 4, 8, 16$, the main misdetections should come from $T2$, so the explanations will be similar to these of Section *V.B*.

C. *Classification Performance*

Tab. II shows the confusion matrix of best CCR performance in classification over all the implementations, which is KNN with $n = 6, k = 97$. As can be seen, $T2$ is the most difficult attack to classify, which was detected as normal in 6.39% of the cases. For all the experiments, the overall misclassification rates are always lower than 4.7%.

When $T0$ are removed from the dataset, our models classify correctly $T1$, $T4$, and $T16$ in most of the cases. The main confusions are those between $T2$ and $T8$. Tab. III shows the confusion matrix between $T8$ and $T2$ when using the SVM *fitcecoc*. Our approach achieved small misclassified rates at less than 0.6%, which is a significant improvement from [1] where about 20% of $T2$ are misclassified as $T4$ or $T8$. The confusion rates also decrease when n increase. $MTDT$ is the main reason for this improvement. As $T2$ is a translated form of a legitimate trajectory, $MTDT(T2)$ value should be small. Meanwhile, $MTDT(T8)$ values should be much larger than $MTDT(T2)$ because $T8$ will have high deviations from any translated trajectories. These deviations will also be accumulated to be higher when n increases, which helps to classify between $T2$ and $T8$ better.

D. Inspection Length Considerations

As can be seen from previous discussions, a small n can give high precision rates in most of the cases, however, the prediction models will be more sensitive, which leads to lower recall rates. Moreover, too small n can also lead to misclassification between $T2$ and $T8$. On the other hand, a large n provides high recall rates and help to classify the attacks correctly. However, it will also decrease the precision rates significantly, not mention that in reality, long observations of the suspected vehicles are more difficult to achieve. Therefore, for a balance between the detection and classification capability and the potential to apply in real life scenarios, the recommended value of n to choose is 5.

VI. Conclusion

In this paper, we introduced three novel features extracted from an $n-sequence$ trajectory for use in machine learning models to detect and classify misbehaviour attacks. We implemented these features in KNN and SVM and compared them to a previous approach that used different features. We evaluated the detection performance of our models using the VeReMi dataset and achieved overall precision of up to 99.7% while maintaining a recall in excess of 99%. The classification accuracy also outperforms the previous best obtained results with less than a 4.7% misclassification rate. Moreover, by studying the impacts of observation length on our models' performance, we showed that our approach can provide reliable judgements even after as few as 3 observations (i.e. 3 seconds), which suggests that they can be useful for near real-time detection and classification. In the future, we aim to extend the attack models (e.g. adding more sophisticated types of misbehaviours) and improving the attack detection techniques to be able to detect and classify more types such as Sybil, replay, or stealthy attacks.

Acknowledgement**.** This work was funded by UK Research and Innovation through INNOVATE UK in project CAPRI (TS/P012264/1).

References

[1] S. So, P. Sharma, and J. Petit, "Integrating Plausibility Checks and Machine Learning for Misbehavior Detection in VANET," in *2018 17th IEEE International Conference on Machine Learning and Applications (ICMLA)*, 2018, pp. 564-571.

[2] C. Maple, "Security and privacy in the internet of things," *Journal of Cyber Policy,* vol. 2, no. 2, pp. 155-184, 2017/05/04 2017.

[3] R. W. v. d. Heijden, A. Al-Momani, F. Kargl, and O. M. F. Abu-Sharkh, "Enhanced Position Verification for VANETs Using Subjective Logic," in *2016 IEEE 84th Vehicular Technology Conference (VTC-Fall)*, 2016, pp. 1-7.

[4] R. W. v. d. Heijden, S. Dietzel, T. Leinmüller, and F. Kargl, "Survey on Misbehavior Detection in Cooperative Intelligent Transportation Systems," *IEEE Communications Surveys & Tutorials,* vol. 21, no. 1, pp. 779-811, 2018.

[5] R. W. van der Heijden, T. Lukaseder, and F. Kargl, "VeReMi: A Dataset for Comparable Evaluation of Misbehavior Detection in VANETs," Cham, 2018, pp. 318-337: Springer International Publishing.

[6] C. Sommer, R. German, and F. Dressler, "Bidirectionally Coupled Network and Road Traffic Simulation for Improved IVC Analysis," *IEEE Transactions on Mobile Computing,* vol. 10, no. 1, pp. 3-15, 2011.

[7] H. Stübing, J. Firl, and S. A. Huss, "A two-stage verification process for Car-to-X mobility data based on path prediction and probabilistic maneuver recognition," in *2011 IEEE Vehicular Networking Conference (VNC)*, 2011, pp. 17-24.

[8] C. Yavvari, Z. Duric, and D. Wijesekera, "Vehicular dynamics based plausibility checking," in *2017 IEEE 20th International Conference on Intelligent Transportation Systems (ITSC)*, 2017, pp. 1-8.

[9] J. Firl, *Probabilistic Maneuver Recognition in Traffic Scenarios.* KIT Scientific Publishing, 2015.