Federated Learning-based Scheme for Detecting Passive Mobile Attackers in 5G Vehicular Edge Computing

Abdelwahab Boualouache* and Thomas Engel
FSTM- Faculty of Science, Technology and Medicine
SnT - Interdisciplinary Centre for Security, Reliability and Trust
University of Luxembourg, Esch-sur-Alzette, L-4365, Luxembourg
Email: {abdelwahab.boualouache, thomas.engel}@uni.lu

***Abstract*—Detecting passive attacks is always considered difficult in vehicular networks. Passive attackers can eavesdrop on the wireless medium to collect beacons. These beacons can be exploited to track the positions of vehicles not only to violate their location privacy but also for criminal purposes. In this paper, we propose a novel federated learning-based scheme for detecting passive mobile attackers in 5G Vehicular Edge Computing. We first identify a set of strategies that can be used by attackers to efficiently track vehicles without being visually detected. We then build an efficient Machine Learning (ML) model to detect tracking attacks based only on the receiving beacons. Our scheme enables Federated Learning (FL) at the edge to ensure collaborative learning while preserving the privacy of vehicles. Moreover, FL clients use a semi-supervised learning approach to ensure accurate self-labeling. Our experiments demonstrate the effectiveness of our proposed scheme to detect passive mobile attackers quickly and with high accuracy. Indeed, only 20 received beacons are required to achieve 95% accuracy. This accuracy can be achieved within 60 FL rounds using 5 FL clients in each FL round. The obtained results are also validated through simulations.**

***Index Terms*—5G Vehicular Edge Computing; Machine Learning, Federated learning, Security, Privacy, Passive Attacker Detection.**

I. Introduction

We are at the dawn of a new era of vehicular networks. Empowered by Artificial Intelligence (AI), connected vehicles are currently self-driving [1]. In addition, by bringing the cloud storage and processing capacities to the edge of the network, the 5G-enabled Vehicular Edge Computing paradigm has emerged [2]. This paradigm is addressing the limitations of traditional vehicular networks and enables ultra-low-latency and high bandwidth Vehicle-to-everything (V2X) applications. Moreover, combined artificial intelligence with edge computing is bringing new sophisticated security and privacy solutions for connected vehicles [3].

Connected vehicles frequently send safety messages, called beacons or CAMs (Cooperative Awareness Messages). These beacons are sent in clear-text and contain sensitive mobility information of vehicles such as its identifier, position, speed, and acceleration [4]. The purpose of these messages is to provide awareness for vehicles about their surrounding environment. However, these messages can be easily collected and exploited by passive attackers. Unlike active attackers that can alter or inject false messages, passive attackers can only eavesdrop on beacons for position tracking to know every location visited by vehicles. The motivations of passive attackers are ranging from curiosity and privacy violation of drivers to tracking from criminal purposes [5]. For thwarting tracking attacks, Standards Developing Organizations (SDOs) recommend that connected vehicles use sets of pseudonyms instead of fixed identifiers while broadcasting their safety messages [6, 7]. Pseudonyms are generally temporal identifiers whose vehicles change them regularly to ensure the unlinkability between their messages [7]. However, several studies have demonstrated that a simple pseudonym change is ineffective against a global passive attacker who eavesdrops every message in the network [8]. For enforcing protection against this attacker several pseudonym-changing strategies have been proposed [9]. However, effective strategies have many negatives impacts on safety-related applications [10]. On the other hand, the assumption of a global passive attacker is not realistic since the global coverage requires the deployment of a large number of static sniffing stations, which not only generates significant costs and efforts but also makes them visually detected by authorities. Thus, a simple approach is to use passive mobile attackers instead. These attackers can be deployed on connected vehicles, which are already part of the system, making them more difficult to detect than static attackers. Unlike tracking using cameras [11], passive mobile attackers rely on communication to track vehicles and can employ a range of strategies to avoid being visually detected. However, these strategies are not yet defined.

Recently, vehicular networks have benefited from the advances in machine learning in the areas of network security. Indeed, several ML-based Misbehavior Detection Systems (ML-based MDSs) have been proposed for the efficient detection of internal attackers [12–20]. However, existing solutions have only been proposed to detect active attackers since detecting passive attacks is usually reported as impossible [21]. In addition, most of the existing ML-based MDSs are adopting a centralized learning approach to train their ML models, and no updates are provided after deploying the model, which limits the performance of these systems to enhance their accuracy and detect unseen misbehaviors. Moreover, ML-based MDS supporting ML model updates either (i) violates privacy

preservation of vehicles since multi parts share training data sets [18, 19], or (ii) generates large overhead since distributed learning requires large signaling overheads between peers to achieve synchronization of ML Model updates[20].

To address the aforementioned issues, in this paper, we propose a novel federated learning-based scheme for detecting passive mobile attackers in 5G-enabled vehicular edge computing. Our scheme defines strategies used by passive mobile attackers to avoid being visually detected by their targets. It also leverages an efficient position-based feature extraction method to propose an accurate ML model for detecting passive mobile attackers. Moreover, our scheme supports federated collaborative learning enabling continuous enhancing of the accuracy of the global model while preserving the privacy of vehicles. On the one hand, FL servers are deployed on geographically distributed edge nodes and employ the Federated averaging (FedAvg) algorithm to calculate the global model weights. On the other hand, FL clients self-label unseen data using a semi-supervised learning approach that is based on the Gaussian Mixture algorithm and takes as input already labeled data. Experiments and simulation demonstrate that our scheme can achieve high accuracy within small FL rounds.

The main contributions of this paper can then be summarized as follows:

- Based on federated learning, we propose a secure and privacy-preserving architecture for detecting passive mobile attackers in 5G-enabled vehicular fog computing.
- We propose a set of strategies that can be used by passive mobile attackers for tracking their targets while avoid being visually detected.
- We propose an algorithm for generating synthetic data characterizing passive mobile attacks and an efficient position-feature extraction method.
- To enable privacy-preserving collaborative learning, we propose an accurate FL-based model for detecting passive mobile attackers. This model leverages a semi-supervised method to self-label data at FL clients and Federated averaging (FedAvg) algorithm to calculate the global model weights at FL servers.
- We carry out an extensive set of experiments to evaluate the detection accuracy of our scheme and validate the results through simulations.

The remainder of this paper is organized as follows. Related works are described in Section II. The proposed federated learning-based architecture for detecting passive mobile attackers is presented in Section III. Section IV describes different strategies used by passive mobile attackers. Section V presents the algorithm proposed to generate synthetic data sets and the method used to extract features. Section VI describes the self-labeling and federated learning processes. Experiments and obtained results are presented in Section VII. Section VIII discusses the obtained results. Finally, a conclusion is given in Section IX.

II. Related Work

A. Passive attacker models

The complexity, heterogeneity, and large-scale of the V2X communication system make it vulnerable to various types of attackers. Understanding attacker models helps to develop efficient attack detection and protection mechanisms. Raya et al. [22] have identified four criteria to classify V2X's attackers: (i) Coverage (global vs. local): unlike a local attacker, a global attacker has an overall coverage of the V2X system. It can then eavesdrop every message broadcast by any vehicle, (ii) Activeness (active vs. passive): an active attacker can alter or inject messages, while a passive attacker can only eavesdrop messages, (iii) Privileges (internal vs. external): an internal attacker is an authenticated member in the V2X system while an external attacker is considered as an intruder; and (iv) intention (malicious vs. rational): while rational attackers aim to achieve their interests, malicious attackers aims to destroy the networks.

A global passive attacker is usually assumed to study the location privacy in connected vehicles. Emara et al. [23] archived 90% of tracking success employing a simple tracking method called the Nearest Neighbor Probabilistic Data Association (NNPDA) [24]. Wiedersheim et al. [8] achieved almost 100% tracking success considering an advanced tracking method called Multi-Hypothesis-Tracking (MHT) [25] incorporating with the Kalman filter. Feiri et al. [26] pointed out that due to the cost of eavesdropping the global coverage is hard to be achieved. Petit et al. [27] defined a new passive attacker model called mid-sized adversary. The coverage of this attacker is in between a local passive adversary and a global passive adversary. Their real-world experiments have demonstrated that the tracking success achieves 90% if only 8 sniffing stations are used. Buttyán et al. [28] have also achieved 90% of tracking success by deploying sniffing stations only at road intersections and used a Bayesian decision algorithm as a tracking method. However, all of these studies are based on the deployment of easily detectable static sniffer stations. On the contrary, passive mobile attackers are difficult to detect because they exploit connected vehicles, which are part of the system.

B. ML-based misbehavior detection systems

Several ML-based misbehavior detection systems have recently been proposed for detecting internal attacks. In the following, we describe the most relevant misbehavior detection systems. Gyawli et al. [12] proposed an ML-based Misbehavior System (MDS) for the detecting false alert and position falsification attacks. The proposed MDS used a supervised learning approach for building ML-models. For detecting false alert attacks, a binary classifier was trained based on data sets generated from emulating this attack on a simulator. Moreover, for detecting false position attacks, a multi-class classifier was trained based on VeRiMi data set [29]. Traditional classification algorithms were used to train the ML-models. The classifiers were built offline and deployed on each connected vehicle for attack detection. Quevedo et al. [13] proposed an ML-based misbehavior detection system for detecting Sybil attacks. A supervised learning approach was also adopted for training ML-models. The training and detection of the attack were performed on edge nodes. The collected data consists of a set of matrices describing the driving patterns of vehicles. The

columns of matrices are the features considered for learning. Each row of a matrix contains driving information of a vehicle at time t. An unsupervised learning data dimensionality reduction technique was used to reduce the dimensions of matrices. Extreme Machine Learning (EML) was used for classification based on data set generated using SUMO mobility generator. Le and Maple. [14] proposed a supervised learning approach to detect false position attacks. This approach compares the trajectory of vehicles with trajectories of legitimated vehicles. Three features were proposed to compare the trajectories. Base on the proposed features, a multi-class classifier was trained on VeRiMi data set [29] to detect the five position-falsification attacks. Two classifier algorithms (Support Vector Machine (SVM) and k-Nearest Neighbors (KNN)) were tested based on a MATLAB implementation. Tan et al. [15] proposed an unsupervised learning approach to detect denial of service attacks (DoS). RSUs (Road Side Units) collect the traffic flows of vehicles, which is defined as a sequence of packets from the source to the destination. Each flow contains σ packets with the corresponding time series. The Agglomerate Hierarchical clustering was applied for create clusters of similar traffic flows. In each step of the clustering algorithm, the dynamic time wraping distance [30] was used to calculate distance between the time series of different traffic follow. Alheeti et al. [16] proposed a supervised learning approach to detect Grey hole and rushing attacks in connected vehicles. The data sets were generated by simulating Grey-hole and rushing attacks with an adapted version of AODV protocol on a network simulator. 15 features were selected to train a binary classifier for each attack. Two classification algorithms were used for training: SVM, and Artificial Neural Network (ANN). Kim et al. [17] proposed an ML-based MDS for software-defined connected vehicles where vehicles analyze the incoming traffic and forward some selected data flows to the Software Defined Networking (SDN) controller. Based on these data flows, the SDN controller trains a multi-class classifier using the SVM classification algorithm. The parameters of the trained model are forwarded to vehicles for using it in the detection of misbehaving vehicles. Negi et al. [18] proposed an anomaly detection system for connected vehicles. The proposed system leverages on unsupervised learning approach based on Long short-term memory (LSTM). The speed up the training of the model over big data, the training was performed in a cluster of servers instead of one server. Once the anomaly detection model was trained, the parameters on this model are distributed to vehicles for the real-time detection of anomalies. To keep the model updated, the model was retrained over time based on data newly collected and the parameters of the new model are distributed to vehicles. Li et al. [19] proposed an ML-based misbehavior detection system based on the transfer learning approach. The idea is to transfer the knowledge acquired building ML-models on a large on labeled data to build ML-model with a small labeled data. This paper proposed two solutions to update the ML-model: (i) Cloud assisted approach where the unlabeled data is sent to the cloud for training and updating the ML-model, and sending it back to vehicles (ii) local update where each vehicle updates its ML-model based on a pre-training model to assign pseudo-labels to data. This pseudo-label data are then used after that to transfer learning. Their experiments showed how to exploit knowledge built from detecting injection and impersonation attacks for detecting flooding attacks. Zhang et al. [20] proposed an ML-based misbehavior detection scheme of network-level attacks based on a distributed machine learning approach. The authors supposed that each vehicle has its labeled data. The global ML-model is collaboratively building between vehicles without exchanging data sets between them. This scheme applies an Alternating Direction Method of Multipliers (ADMM) to a class of empirical risk minimization (ERM) problems for training the global ML-model. Instead of sharing the data sets, the vehicles share only updates of the loss functions.

Table I compares our scheme with related ML-based misbehavior detection schemes. Unlike the existing schemes, our scheme is focusing on detecting passive mobile attackers. It combines a semi-supervised approach for self-labeling data in FL clients and the supervised learning for training and updating the global ML model. Moreover, ML inference location in our scheme is connected vehicles, since connected vehicles are responsible for detecting and reporting passive mobile attackers. On the other hand, our scheme adopts a federated learning model, which, unlike the existing schemes [18, 20], allows updating the global ML-model with ensuring the privacy preservation and small overhead.

C. Federated Learning

Federated learning (FL) is a distributed machine learning approach that enables collaboration between multiple parties to jointly train a global model without sharing their data sets, which mitigates privacy risks [31]. In the FL architecture [32, 33], multiple FL clients cooperate with the FL server to train a global ML-model. The training of the global model is carried out within multiple rounds until the FL server achieves a satisfactory global model. In each round, the FL server selects a set of FL clients and sends them the model obtained after the last round. Based on the received model, each FL client uses its local labeled data set to calculate its local updates of the global model using Stochastic Gradient descent (SGD). After the end of the round, all the selected FL clients send their local updates to the FL server. Once all the updates are received, the FL server uses the Federated averaging (FedAvg) algorithm to aggregate local updates for calculating the global model. Federated learning has recently been exploited in several V2X applications. Samarakoon et al. [34] used the FL to estimate the tail distribution of the network-wide queue lengths. Lu et al. (1) [35] proposed an FL scheme for resource sharing in vehicular networks. Liu et al. [36] proposed a FL scheme for traffic flow prediction in vehicular networks. Lu et al. [37] discussed the benefits of data sharing among vehicles for collaborative analysis in vehicular networks. Lu et al. (2) [38] proposed a privacy-preserving federated learning architecture for enhancing data privacy. They also designed a two-phase mitigating scheme consisting of intelligent data transformation and collaborative data leakage detection. Yeet al. [39] discussed the use of FL for image classification in vehicular Edge Computing. Our scheme

TABLE I: A comparison of our scheme with relevant state-of-the-art schemes.

Solution	Attacks	ML-type & Design	Learning Model	ML-Inference location	Model update	Privacy preserving	Overhead
Gyawli et al. [12]	False alert Position falsification	Supervised Learning	Centralized learning	Connected vehicle	No	No	Large
Quevedo et al. [13]	Sybil	Supervised learning	Centralized learning	Edge	No	No	Large
Le and Maple [14]	Position falsification	Supervised learning	Centralized learning	Connected vehicle	No	No	Large
Tan et al. [15]	DoS	Unsupervised learning	Centralized learning	RSU	No	No	Large
Alheeti et al. [16]	Grey hole and rushing	Supervised learning	Centralized learning	Connected vehicle	No	No	Large
Kim et al. [17]	Network-level	Supervised learning	Centralized learning	Connected vehicle	No	No	Large
Negi et al. [18]	Anomalies	Unsupervised learning	Data center distributed learning	Connected vehicles	Yes	No	Large
Li et al. [19]	Network-level, Flooding impersonation data injection	Transfer learning Supervised learning	Centralized learning	Connected vehicle	No	No	Large
Zhang et al. [20]	Network-level	ADMM Supervised learning	Peer-to-peer distributed learning	Connected vehicle	Yes	Yes	Large
Our Scheme	**Passive mobile**	**Supervised learning Semi-supervised learning**	**Federated Learning**	**Connected vehicle**	**Yes**	**Yes**	**Small**

considers an edge layer consisting of multiple nodes, where each node controls a specific geographic region. Mowlaet al. [40] proposed an FL based jamming attack detection mechanism for flying ad hoc networks. Unlike [40], our exploit federated learning for detecting passive mobile attacks in 5G vehicular edge computing.

Table II presents the abbreviations and notations used throughout the paper.

TABLE II: Abbreviations and notations used throughout the paper.

Notation	Description
AI	Artificial Intelligence
$V2X$	Vehicle-to-everything
DoS	Denial of Service
ANN	Artificial Neural Network
SDN	Software Defined Networking
FL	Federated Learning
ML	Machine Learning
MDS	Misbehavior Detection System
$AODV$	Ad-hoc On-demand Distance Vector
SVM	Support Vector Machine
RSU	Road-Side Unit
SGD	Stochastic Gradient Descent
CR	Communication Range
SD	Safety Distance
CD	Caution Distance
RD	Reaction Distance
P	Period
KNN	k-Nearest Neighbors
$LSTM$	Long short-term memory
ELM	Extreme Learning Machine
CA	Certifications Authority
$FLS-DS$	FL Server Data Set
$L-UDS$	Local Unlabelled Data Set
$L-LDS$	Local Labelled Data Set

III. System Model

As illustrated in Figure 1, we consider a federated learning-based architecture for 5G-enabled vehicular edge computing consists of two layers. The infrastructure layer includes vehicles and base stations equipped with V2X communication technologies. In this layer, communications are multi-hop Vehicle-to-Vehicle (V2V). Vehicle-to-Infrastructure (V2I) communications are used to communicate with the 5G-Edge

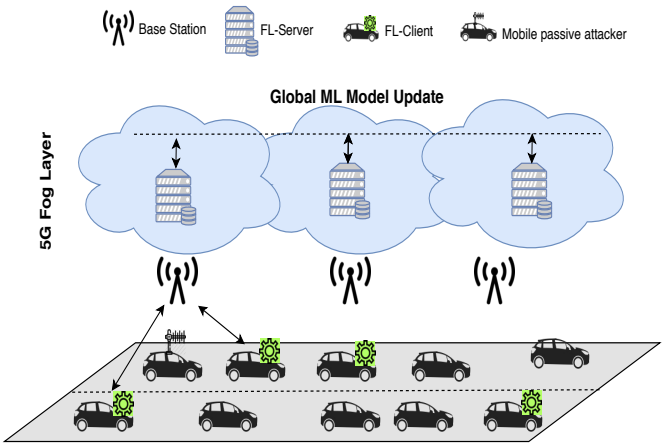


Fig. 1: Federated learning-based architecture for detecting passive mobile attackers in 5G Vehicular Edge computing

Layer. Each vehicle periodically broadcasts a safety message every t millisecond, where each message includes a pseudonym, a position, a timestamp, the velocity, and content. Since these messages are sent in clear, the passive mobile attacks can easily exploit them to track the positions of connected vehicles as described in Section IV.

The 5G Edge Layer mainly consists of FL servers connecting through secure 5G links for sharing the global ML-model updates. Each FL server manages a limited geographic zone and communicates with selected FL clients (vehicles) within this zone thorough secure 5G links. During the initialization process, the Certification Authority (CA) delivers to vehicles different credentials such as public and private keys, and certificates. Vehicles use these credentials to ensure the authentication/integrity of their messages and encrypt them (if necessary). The interaction between an FL server and an FL client is illustrated in Figure 2. The FL server has two modules: (i) Model building, and (ii) Model updating. Also, each vehicle has two modules: (i) Passive attacker detection, and (ii) the FL Agent. The FL Agent module is only activated if the FL server selects the vehicle as an FL client. The internal design of the FL agent is described in Section VI.

The model building module uses synthetic labeled data set to build an initial global ML-model. The processes of generating synthetic data set and extracting features are described in subsections V-A and V-B respectively. The model updating module is used to update the global ML-model. In each round, this module selects some vehicles to serve as FL clients and sends the initialization parameters to their FL agent modules. At the end of each round, the FL agents send their updates to the FL server. The updated global model is shared with all connected vehicles and deployed on their passive attacker detection modules. The detection of passive attackers is only based on received beacons. For each received beacon, the features extraction process is applied and the ML model is used to detect and classify passive attackers. Once the attack is detected, the attackers are reported to CA for investigations and taking the required actions.

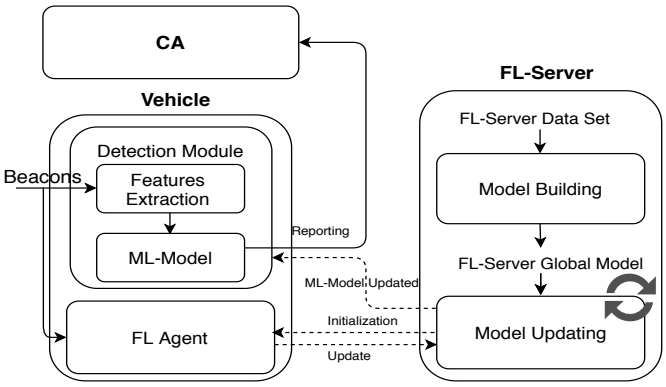


Fig. 2: The interaction between an FL server and an FL client

IV. Passive Mobile Attackers Strategies

Passive mobile attackers use their wireless interfaces to collect safety messages broadcast by vehicles and use tracking methods to link between the messages that belong to their targets. Leveraging messages instead of cameras make tracking much easier for these attackers. The basic approach used by passive mobile attackers is usually to keep himself in the range and a short distance from their targets to ensure the good reception of beacons. However, passive mobile attackers can use a set of strategies for preventing to be visually noticed by the target. In the following, we identify strategies that can be used by passive mobile attackers. These strategies are classified into two categories: Single attacker and Collaborative attackers.

1) Single Attacker: In this category, only one single passive mobile attacker is used to track the vehicle. Figure 3 (a) illustrates an example of this attacker. he attacker (the red vehicle) always keeps, at least, a caution distance between itself and the target (the blue vehicle) and still inside the attacker zone. This zone begins at the end of the caution distance and ends at the maximum communication range of the target. Inside this zone, several strategies can be applied by the attacker to prevent to be visually noticed and detected by the target. These strategies are listed below:

- **Constant Strategy:** In this strategy, the attacker tries to keep exactly a caution distance between itself and the target. This can be done by adapting its speed according to the speed of the target.
- **Attack-and-stop strategy**: In this strategy, instead of doing the tracking all the time, the attacker periodically alternates between attack and no-attack modes. In the attack mode, the attacker applies the constant strategy while in the no-attack mode the attacker acts as an ordinary vehicle.
- **Random strategy**: This strategy is a more advanced form of attack-and-stop strategy. The same as the attack-and-stop strategy, the attacker alternates between attack and no-attack modes. However, in this strategy, the alternation occurs randomly and not periodically.

2) Collaborative attackers: Unlike the first category, in this category, multiple attackers collaborate for tracking of the target. These attackers can synchronize and communicate between them to accurately track the target and reduce the probability of being visually detected. Each attacker of the group of attackers can apply one of the strategies previously mentioned in the first category. Depending on the environment, two tracking collaborative strategies can be applied:

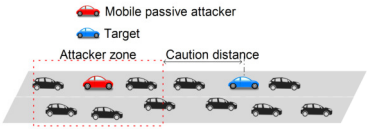
- **Urban strategy**: This strategy is convenient for the urban environment, which is usually characterized by the existence of road intersections. As shown in Figure 3 (b) collaborative attackers are deployed along with road segments. Each attacker is used to track the target only in one road segment.
- **Highway strategy**: This strategy is convenient for a highway environment where several attackers alternate to track the target. Figure 3 (c) illustrates the highway strategy. Although three attackers are tracking the target, they synchronize between them to ensure that only one attacker is activated on one time to avoid their detection.

V. Data Set and Feature Extraction

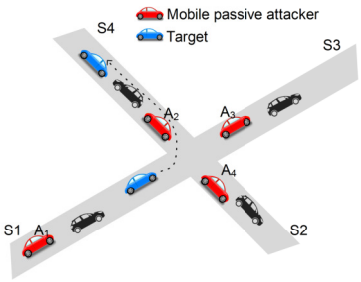
This section describes the process of generating synthetic data for training and testing the initial global model. It also describes how the features are extracted for providing fast and accurate attack detection. The generation of synthetic data is based on a set of algorithms emulating normal behavior and attacker behaviors. The use of synthetic data allows developing more accurate ML-models since synthetic data can encompass all the envisioned scenarios [41]. Indeed, it is hard to cover all the possible cases by simulations or by experiments [42]. On the other hand, for the feature exaction, we use an inter distance-based approach to extract features from the synthetic data and build a data set.

A. Synthetic Data Generation

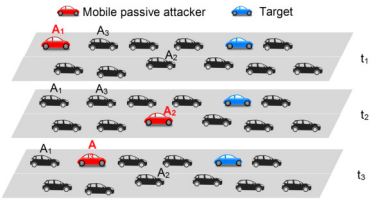
The generation of synthetic data is based on the distance between the target and the attacker. In practice, each vehicle periodically stores the distance between itself and each of the neighboring vehicles. Figure 4 illustrates the parameters used to generate the synthetic data: (i) the maximum communication range (CR); (ii) the Safety Distance (SD); (iii) the Caution Distance (CD), which is the distance that the attacker keeps avoiding visual detection by the target; and (iv) the Reaction



(a) Single attacker



(b) Collaborative attackers: Urban strategy



(c) Collaborative attackers: Highway strategy

Fig. 3: Passive attacker strategies

Distance (RD), which is an additional margin of distance that the attacker may take while it is trying to adapt its speed with the attacker's speed to maintain the CD. This mainly occurs when encountering obstacles like other vehicles.

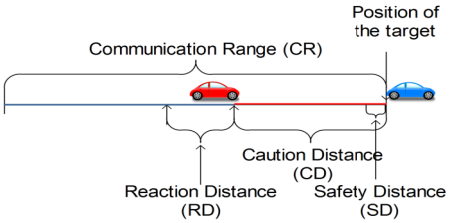


Fig. 4: Parameters of synthetic data

Algorithm 1 shows the pseudo-codes of the algorithms used to generate synthetic data. n is the number of normal vehicles, m is the number of taken measures. n_c is the number of constant attackers, n_{as} is the number of attack-and-stop attackers, P is a period to switch between attack and no-attack in the attack-and-stop strategy, n_r is the number of random attackers, n_{cl} is the total number of collaborative attackers and n_g is the size of a group of collaborative attackers.

- **Subroutine 1** is used to generate a trace of normal vehicles. The variations of the inter-distance between vehicles are random and can range from the limit of SD to the limit of CR.
- **Subroutine 2** is used to generate a trace of constant attackers. As already mentioned this attacker tries to keep a CD to avoid being noticed by the target. It always tries to adapt its speed within the reaction zone to maintain the CD.
- **Subroutine 3** is used to generate a trace of attack-and-stop attackers. This attacker periodically alternates between attack and normal. P is the period used to switch between the normal and the attack modes.
- **Subroutine 4** is used to generate a trace to random attackers. Unlike the attack-and-stop attacker, the attacker randomly switches the normal and the attack modes.
- **Subroutine 5** is used to generate a trace of collaborative attackers. These attackers are organized in groups. Attackers of each group synchronize between them, where each attacker uses a single attack strategy. For example, in the pseudo-code, collaborative attackers use a constant strategy. The pseudo-code applies the highway strategy since the urban strategy consists of single attackers geographically distributed, which can be generated using the previous subroutines.

B. *Feature selection*

To use machine learning, we need to extract features that contain information on the variation of inter-distance between vehicles. Figures 5 illustrates the variation of the inter-distance between a vehicle and its neighbors. Each curve shows the inter-distance between the vehicle and a given neighbor. The black curves are normal while the red curves are suspicious since the inter-distances are stable, which can be interpreted as constant attackers.

In our scheme, we use a signal sampling technique to extract features representing the variation of inter-distances between vehicles. Specifically, this technique converts the inter-distance trace to a sequence of samples where each sample serves as one feature for the data set. Figure 6 shows an example of the process of feature extraction from the inter-distance trace. The sampling process is monitored by two parameters: the sampling length (Δ) and the sampling interval (δ). First, It divides the trace into δ fragments. Then, it calculates inter-distance (x_i) for each fragment. The obtained consecutive Δ samples represent the feature vector $X = [x_{i+1}, x_{i+2}....., x_{i+j}, ..., x_{i+\Delta}]$. As in signal processing, the smaller the sampling interval is, the higher fidelity the features can achieve to represent the original signal, and, in this case, the inter-distance variation.

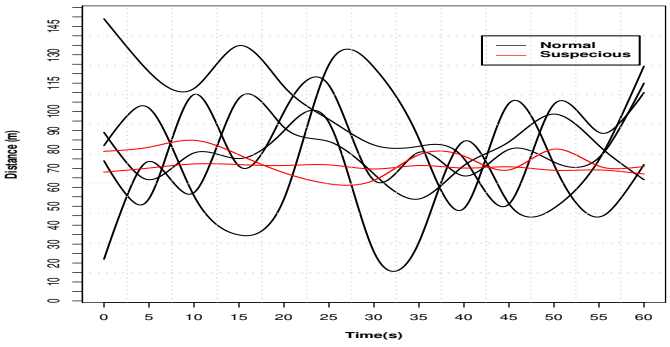


Fig. 5: The variation the distance between a vehicle and its neighbors

Algorithm 1: Generation Synthetic Data

Initialize
  n, m, n_c, n_{as}, P, n_{cl}, n_g, CR, SD, CD, RD;
Subroutine 1 — Generate normal traces
  Data: n, m, CR, SD
  Result: Normal attackers data (N)
  while $i < n$ **do**
    while $j < m$ **do**
      N [i,j] ← SD + Rand()% (CR-SD) ;
    end
  end
Subroutine 2 — Generate constant attackers traces
  Data: n_c, m, CD, RD
  Result: Constant attackers data (C)
  while $i < n_c$ **do**
    while $j < m$ **do**
      C [i,j] ← CD + Rand() % RD ;
    end
  end
Subroutine 3 — Generate attack-and-stop attackers traces
  Data: n_{as}, m, CD, RD, P
  Result: Attack and stop attackers data (AS)
  while $i < n_{as}$ **do**
    while $j < m$ **do**
      if *if ((j/P) % 2)==0* **then**
        AS [i,j] ← CD + Rand() % RD ;
      else
        AS [i,j] ← CD + Rand() % (CR-CD) ;
      end
    end
  end
Subroutine 4 — Generate random attackers traces
  Data: n_r, m, CD, RD
  Result: Random attackers data (R)
  while $i < n_r$ **do**
    while $j < m$ **do**
      if *if (Rand () % 2)==0* **then**
        AS [i,j] ← CD + Rand() % RD ;
      else
        AS [i,j] ← CD + Rand() % (CR-CD) ;
      end
    end
  end
Subroutine 5 — Generate collaborative attackers traces
  Data: n_{cl}, n_g ,m, CD, RD,SD,CR
  Result: Collaborative attackers data (CL)
  g=nb_{cl} / nb_g ;
  while $i < g$ **do**
    while $k < nb_g$ **do**
      while $j < m$ **do**
        if *if ((j >= k ∗ (m / nb_g)) and (j < ((k+1) ∗ (m / nb_g))))* **then**
          CL [i,j] ← CD + Rand() % RD ;
        else
          CL [i,j] ← SD + Rand()% (CR-SD) ;
        end
      end
    end
  end

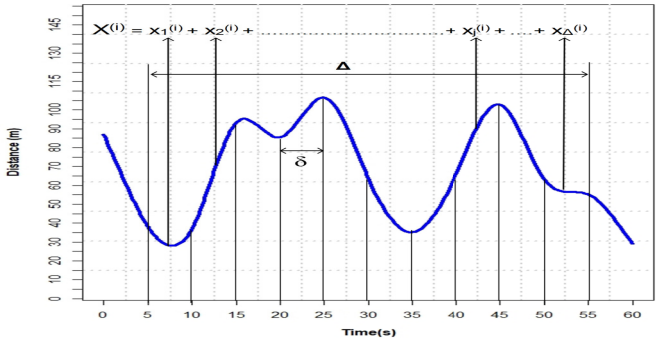


Fig. 6: Trace sampling

VI. Self-labeling and Federating Learning

In this section, we describe the process of updating the global model. The periodic updates of the global model enhance the detection capabilities of our scheme. Thanks to federated learning, these updates are performed with a small overhead and are protected from privacy risks. Figure 7 illustrates the internal design of an FL agent and the process of updating the global model. As we can see the FL agent has two internal components: self-labeling and model update. The FL agent has also two operating modes: passive and active. In the passive mode, the FL agent keeps collecting beacons and links them with their corresponding pseudonyms. If the number of linked beacons of a given pseudonym is enough, the FL agent applies the feature extraction process as described in subsection V-B and stores the feature vector in a database of unlabelled data set. When the FL server selects the connected vehicle as an FL client, its FL agent will turn to the active mode. In this mode, the FL Agent starts interacting with the model update module. It will then receive an initial data set and current parameters of the global model from this it. The received data set will be used by the self-labeling component to label the unlabeled data as described in subsection VI-A. Once the data is labeled, the model update component takes it with received parameters as input to update the global model. The whole process update of the global model is described in subsection VI-B. These updates are sent to the FL server as soon as finalized. The database of local unlabeled data is periodically purged to save the storage space of the vehicle.

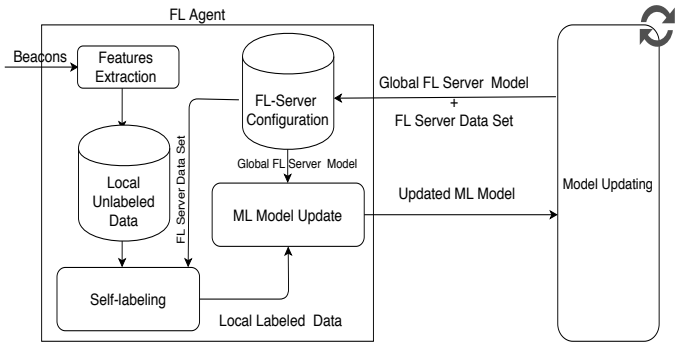


Fig. 7: FL Agent internal design and Global ML-model update process.

A. Self-labelling

In this section, we describe the process of self-labeling. To label the unlabeled data set we use a semi-supervised approach. Figure 8 illustrates the proposed approach. For the visualization purpose, we apply the Principal Component Analysis (PCA). Figure 8 (a) shows the data after applying the Gaussian mixture clustering algorithm on the labeled data received from the FL server. Each cluster corresponds to an attack class. In Figure 8 (b) we combine the labeled data set received from the FL server and unlabeled data locally stored in the FL client and apply the Gaussian mixture clustering algorithm again. In the Figure, the black points are the

unlabeled data that need to be matched to the corresponding classes.

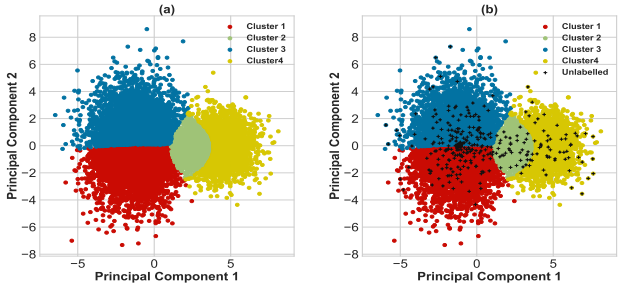


Fig. 8: Semi-supervised learning for self-labelling

To achieve this, we propose a matching algorithm that matches the unlabelled data to the corresponding classes. Algorithm 2 illustrates our proposed self-labeling algorithm. This Algorithm takes the labeled data set received from FL server (FLS-DS) and the Local Unlabeled Data Set (L-UDS) as input and returns the Local labelled Data set (L-LDS) as an output. The algorithm starts by extracting the labels and feature vectors from FLS-DS in steps 5 and 4 respectively. In step 7, the feature vectors of FLS-DS are combined with L-UDS to build a new data set (DS). In Step 8, the Gaussian Mixture clustering algorithm is applied to extract the cluster labels of FLS-DS and L-UDS respectively. Step 9 links the clustering labels and ground truth labels. The mapping function is based on trying all possible rearrangements of the clusters labels to see which of them best fit the ground truth vector. The result of the mapping function is a mapping scheme between the cluster labels and the ground truth labels. In step 10, this mapping scheme is used to find the correct label of each vector of the unlabeled data leveraging on the cluster label.

Algorithm 2: Self-labeling procedure

2 **Self-labelling**
3 ***Input: FL Server Data Set (FLS-DS), Local Unlabelled Data Set (L-UDS)***
4 ***Output: Local Labelled Data Set (L-LDS)***
5 $FLS-DS\_Lbls \leftarrow Get\_labels(FLS-DS)$
6 $FLS-DS\_Features \leftarrow Get\_features(FLS-DS)$
7 $DS \leftarrow$ **Combine** $(L-UDS, FLS-DS\_Features)$
8 $FLS-DS\_C, L-UDS\_C \leftarrow$ **GaussianMixture**(DS)
9 $Map\_sch \leftarrow$ **Mapping**$(FLS-DS\_Lbls, FLS-DS\_C)$
10 $L-LDS \leftarrow$ **Match**$(Map\_sch, L-UDS, L-UDS\_C)$

B. Federated Learning

As already mentioned in subsection II-C, updating the global model is carried out within several rounds. Each round has three main steps: (i) FL client selection, (ii) Local update, and (iii) Global model averaging. To avoid model bias, we propose that, in each round, FL servers select new FL clients, which are not already selected in previous rounds. In addition, we assume that D_k is a local data set of an FL client k and $n_k = | D_k |$. D_k consists of pairs of features and labels, (x_i,y_i), $i = 1, ..., n_k$ for local n_k data points given n global data points trained in the round. This allows each of FL clients locally calculates the weights update of the global model using its local n_k data points as described in subsection VI-B1. Finally, getting weights updates from all FL clients allows the FL server to calculate the updated global model using the federated averaging algorithm as described in subsection VI-B2. Federated learning functions are described in Algorithm 3.

1) Local Update: An FL client k calculates a local update of the model with a finite sum objective of the form,

$$\min_{\omega \in R^d} L_k(\omega) \tag{1}$$

$L_k(\omega)$ is a loss function that is to be minimized with respect to ω. $L_k(\omega)$ is calculated over n_k data points using the formula (2):

$$L_k(\omega) = \frac{1}{n_k} \sum_{i \in D_k} f_i(\omega) \tag{2}$$

where,

$$f_i(\omega) = f(x_i, y_i; \omega) \tag{3}$$

$f_i(\omega) = \ell\ (x_i,y_i,\omega)$ is the loss of the prediction on the data point (x_i,y_i) made with model parameters ω. As shown in **FL-ClientUpdate** function, an FL client (k) splits n_k data points into B sized batches in step 6. The received global model is locally trained on E epochs in steps (7-11). In each epoch e, a local vector of weights $\omega \in R^d$ is updated over all the set of batches b. The update of the vector of weights is performed in step 9, where η is the learning rate and $\Delta\ \ell\ (\omega;$ b) is the gradient of the local objective function of the client k.

Algorithm 3: Federated learning processes

2 **FL-ClientUpdate**(k, ω)
3 ***Input:*** n_k
4 ***Output:*** ω^k
5 Extract n_k feature set (x_i,y_i)
6 $b \leftarrow$ Split data n_k into batches of size B
7 **for** *each local epoch e from 1 to E* **do**
8 **for** $b \in B$ **do**
9 $\omega \leftarrow \omega$ - $\eta\ \Delta\ \ell\ (\omega;b)$
10 **end**
11 **end**
12 return ω^k to server
14 **FL-ServerUpdate** ()
15 ***Input:*** ω_0
16 ***Output:*** ω
17 Initialize ω_0
18 **for** *each round r =1,2,...* **do**
19 K $\leftarrow$ desired number of FL clients
20 **for** *each client* $k \in K$ **do**
21 $\omega^k_{r+1} \leftarrow$ FL-ClientUpdate(k, ω)
22 **end**
23 $\omega_{r+1} \leftarrow \sum_{k=1}^{K} \frac{n_k}{n} \omega^k_{r+1}$
24 $\omega \leftarrow \omega_{r+1}$
25 **end**

2) Global Model Averaging: The global model is aggregated at the FL server, which has the following global model objective:

$$\min_{\omega \in R^d} l(\omega) = \frac{1}{n} \sum_{i=1}^{n} f_i(\omega) \tag{4}$$

where,

$$l(\omega) = \sum_{i=1}^{k} \frac{n_k}{n} L_k(\omega) \tag{5}$$

$L_k(\omega)$ denotes the objective function of an FL client k. Formula 5 gives a weighted average from all of the K FL clients since n_k can vary among the K clients. As shown in **FL-ServerUpdate** function, the FL server starts by initializing the weights of the global model in Step 17. Then, in each r round and for each client $k \in K$, ω_{r+1}^k is updated by the **FL-ClientUpdate(**k**,** ω**)** in step 21. At the end of each round r, the federated averaging algorithm is used to derive the weighted averaging of the aggregated client updates as follows:

$$\omega_{r+1} = \sum_{k=1}^{K} \frac{n_k}{n} \omega_{r+1}^k \tag{6}$$

ω_{r+1} is the global weight at round $r+1$ for a total of K FL clients over a total of n data points.

VII. Performance Evaluation

In this section, we evaluate the performance of our scheme. This section is divided into five subsections. First, we describe the metrics used in our evaluations. Second, based on the synthetic data generated using Algorithm 1, we build and compare two multi-class classifiers for defining the global model to deploy on FL servers. Third, we evaluate the self-labeling algorithm presented in subsection VI-A. Fourth, we evaluate federated learning functions presented in subsection VI-B. Finally, we validate the obtained global Model thorough simulations.

A. Evaluation Metrics

Various evaluation metrics are used in our evaluation. In the following, we give the calculating formula for each metric with a short description:

- Accuracy is the ratio of the correctly detected passive attackers to the total of vehicles.

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{7}$$

- Precision calculates the ratio of correctly detected passive attackers to the total detected passive attackers.

$$Precision = \frac{TP}{TP + FP} \tag{8}$$

- Recall calculates the ratio of correctly detected passive attackers to the total actual passive attackers.

$$Recall = \frac{TP}{TP + FN} \tag{9}$$

F1-score can be interpreted as a weighted average of precision and recall.

$$F1 - score = 2X \frac{Precision * Recall}{Precision + Recall} \tag{10}$$

B. ML-Model Building

In this section, we consider that ML models are centrally trained. In this experiment, we train two multi-class classifiers and evaluate their effectiveness in detecting passive mobile attackers. Both of the two multi-class classifiers consider single and collaborative attacks. However, the first multi-class classifier (MC-1) considers collaborative attackers as groups, while the second multi-class classifier (MC-2) deals with each collaborative attacker individually as a single attacker. To generate data for training and testing these multi-class classifiers, we leverage the algorithm presented in subsection V-A. This Algorithm was implemented using Python 3. Table III presents the parameters used to generate data. By combining these parameters and considering 200 instances of each configuration, we have ended up generating 22000 rows. We have also created a python-based Jupyter notebook for data prepossessing and feature extraction applying the method presented in Section V-B. We initially considered that the size of the feature vector equals 100. For training and testing the multi-classifiers the supervised algorithms implemented in Scikit-learn literary.

TABLE III: Data Set Parameters

Parameter	**Value**
Safety Distance (SD)	3 m
Caution Distance (CD)	{50,60, ..., 150} m
Communication Range (CR)	{400,500} m
Reaction Distance (CR)	10 m
p	5s

Table IV shows the multi-class classification results of MC-1 considering various metrics. The results show that the Random Forest classification algorithm gives almost 100 % accuracy.

TABLE IV: Multi-class classification results of MC-1 (number of features = 100)

Model	**Accuracy**	**Precision**	**Recall**	**F1 score**
Logistic Regression	0.395	0.37	0.4	0.38
KNN	0.46	0.43	0.33	0.34
SVM	0.67	0.67	0.67	0.665
Naive Bayes	0.66	0.65	0.66	0.65
Decision Tree	0.83	0.825	0.83	0.83
Random Forest	0.99	0.99	0.99	0.99

To determine the minimum number of features required to detect passive mobile attackers, we variate the number the features and evaluate the performance of MC-1. Figure 9 illustrate the evolution of the accuracy versus the number of features of the three best performing classification algorithms in our experiment. The obtained results confirm that the

Random Forest classifier performs better whatever the number of features is. However, at least 60 features are needed to achieve 90% accuracy.

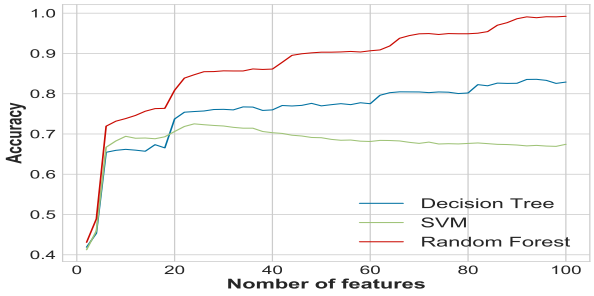


Fig. 9: Accuracy of MC-1 versus the number of features

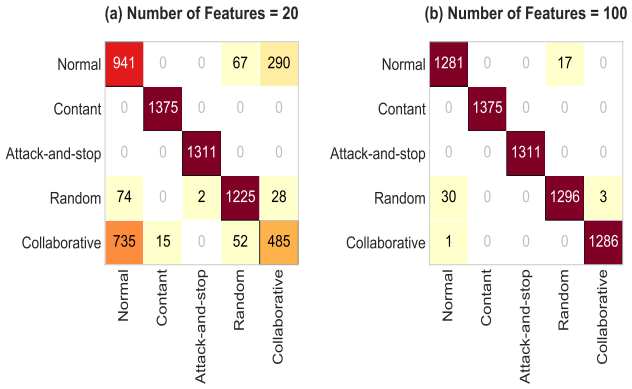


Fig. 11: Confusion matrices

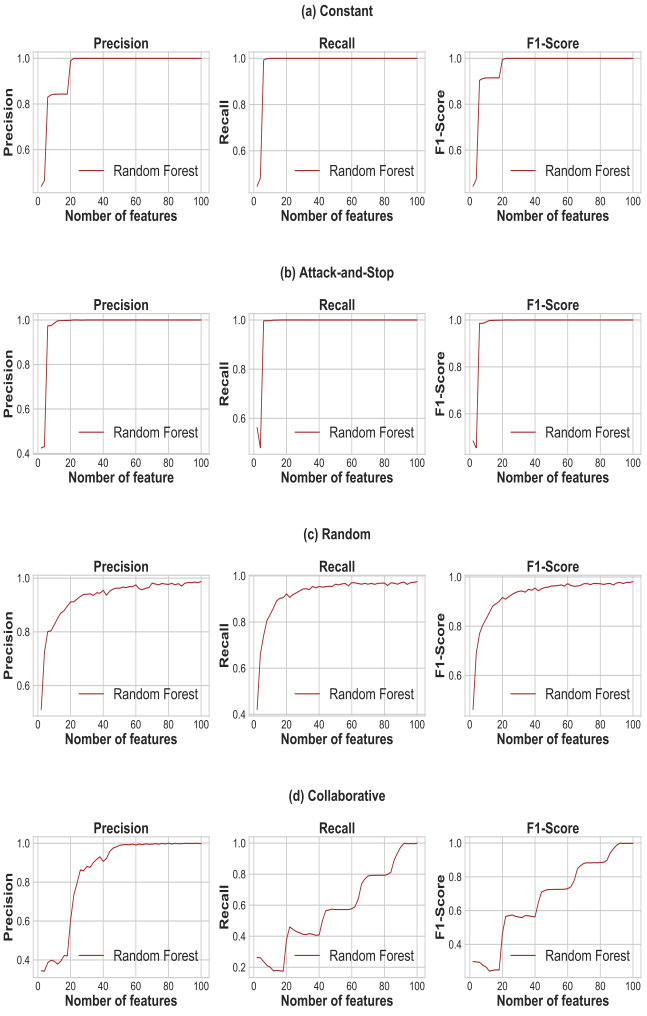


Fig. 10: Precision, Recall, and F1-score of MC-1 versus the number of features

Figure 10 shows the precision, the recall, and F1-score of MC-1 versus the number of features obtained for each attack. Figure 10 (a) shows that, for the constant attack, 20 features are sufficient to obtain almost 100 % of each evaluation metrics. Figure 10 (b) shows that the attack-and-stop is detected faster than the constant attack. Indeed, only 15 features are needed to detect the attack (100 % of each of the evaluation metrics). Figure 10 (c) shows that the random attack is a bit difficult to detect than the two previous attacks. Indeed, with 20 features, we obtain almost 90% of each evaluation metrics. However, this percentage incrementally increases with the increase in the number of features. Figure 10 (d) shows that the collaborative attack is more difficult to detect than the previous attacks (single attacks). Indeed, 85 features are needed to reach 85% of each evaluation metric, and 95 features to reach 100%. The reason for these results can be explained by the confusion matrix shown in Figure 11. As we can see, with 20 features, MC-1 confuses the collaborative with normal behavior and the constant attack. However, with 100 features, MC-1 can accurately distinguish these attacks. The speed of attack detection depends on the number of features. Indeed, the fewer features required to detect the attack, the faster the attack detection. The evaluation results of MC-1 show that single attackers are detected more quickly than collaborative attackers if the latter are viewed as groups. However, since collaborative attackers combine a set of single attackers, we train another classifier (MC-2) that deals with collaborative attackers as single attackers instead of as groups. Table V shows the multi-class results of MC-2 using 20 features. As we can see the random forest classification algorithm also provides the highest results for MC-2. The results show that 97% accuracy is achieved by MC-2 using 20 features, which is better than MC-1 using the same number of features as shown in Figures 12. These results demonstrate that it is better to consider collaborative attackers as single attackers to build the global model for federated learning.

TABLE V: Multi-class classification of all attacks (20 Features) without the collaborative attacks (MC-2)

Model	Accuracy	Precision	Recall	F1 score
Logistic Regression	0.52	0.52	0.52	0.51
KNN	0.79	0.80	0.79	0.77
SVM	0.87	0.87	0.87	0.87
Naive Bayes	0.81	0.81	0.81	0.81
Decision Tree	0.89	0.89	0.89	0.89
Random Forest	0.97	0.97	0.97	0.97

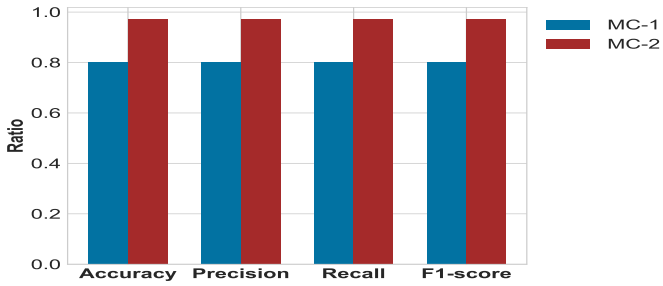


Fig. 12: Comparison between MC-1 and MC-2 (20 features)

C. Self-labeling Evaluation

In this section, we evaluate the self-labeling solution proposed in subsection VI-A. To carry out this evaluation, we have generated novel unlabeled data set using Algorithm 1. As discussed in subsection VI-A, the self-labeling consists of mixing already labeled data with unlabeled data sets to label the unlabeled data set. In Figure 13, we evaluate the ratio of the size of the unlabelled data to the size of the labeled data that allows obtaining good accuracy. Accordingly, we consider different values of ratio ranging from 2% to 30% and measure the accuracy for each ratio. As shown in Figure 13, the best accuracy results are 80% obtained at the ratio 20%.

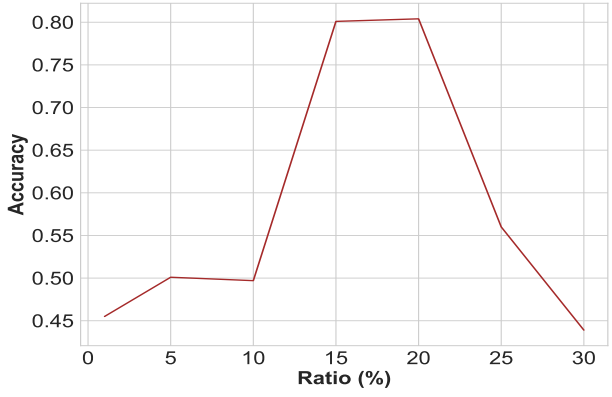


Fig. 13: Accuracy of proposed self-labeling solution versus ratio

D. Federated Learning Evaluation

In this section, we train and test a global model for detecting passive mobile attackers using a federated learning approach as described in subsection VI-B. The federated learning architecture was implemented using Tensorflow and Keras Python libraries. The global model was trained on a single machine equipped with 12 CPU and 15 GO of RAM. FL clients have been implemented as Tensorflow instances running local models. For the multi-class classification, we use a Multi-Layer Perceptron (MLP) model with four layers. The MLP model consists of one input layer and one output layer (softmax is used as an activation function) and two hidden layers (RELU is used as an activation function) each of them has 10 units. To calculate the weights of local models we use the Stochastic Gradient Descent (SGD). The loss function is the categorical cross-entropy and the learning rate equals to 0.01. We also optimize the decay parameter, which controls how the learning time change over time. Indeed, we decay the learning rate with respect to the number of rounds rather than the number of epochs. The Federated Averaging is used after each round to calculate the weight of the global model as described in subsection VI-B. In this evaluation, we consider two data sets (DSs): small Data DS and large DS, which ten times larger than the small DS. We also variate the number of selected FL clients in each round. These values are considered 2, 5, and 10 FL clients.

Figure 14 shows the obtained accuracy values versus the number of rounds. The figure shows, after only 190 rounds, all the configurations achieve more 90% accuracy. It also shows that using a large DS allows getting high accuracy values with fewer rounds than using a small DS. Indeed, more than 93% accuracy is achieved with only 60 rounds using a large DS. Moreover, the Figure shows the number of FL clients selected each round is an influencing factor on accuracy values, and the number round required to archive higher accuracy. For example, 5 FL clients on the large DS, more than 95% accuracy can be achieved in only within 60 rounds while this accuracy cannot be achieved using 2 FL clients in the same number of rounds.

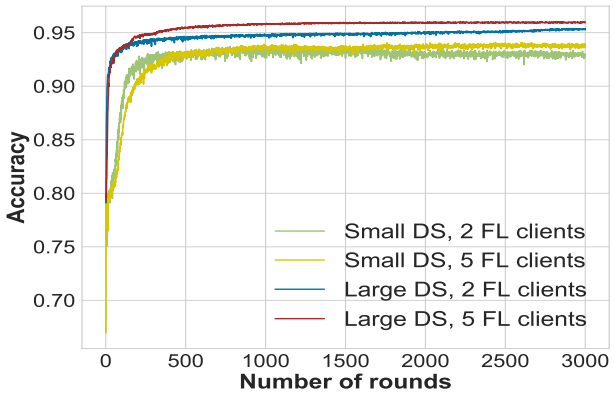


Fig. 14: Accuracy using federated learning

E. Simulations

We have also carried out simulations to validate our proposed model. These simulations are conducted using Veins Simulation Framework [43]. Veins is an inter-vehicular communication simulation framework based on OMNet++ bi-directionally coupled with SUMO road traffic simulation [44]. Table VI summarizes the parameters considered in our simulations.

Parameter	Value
Communication technology	IEEE802.11p
Simulation duration	600 s
Communication Range (CR)	500 m
Beacon Interval	1 s
Number of constant attackers	10
Number of random attackers	10
Number of attack-and-stop attackers	10
Caution Distance (CD)	100 m
Reaction Distance (RD)	10 m
P	30s

TABLE VI: Simulation Parameters

We consider the case of a freeway road. We simulate a 2-lane straight road section of 12 Km. The mobility of vehicles was generated using SUMO. We have considered three scenarios corresponding to the three single passive attacks. In each scenario, ten attackers are trying to track one target. The inter distances between the target were collected. Based on this data, the feature extraction was applied. Only 20 features were used to detect the attack using the model previously developed. Figure 15 shows the accuracy obtained from detecting simulated attacks. We obtain 93% accuracy in detecting all attackers. In particular, attack-and-stop attackers are detected accurately than other attackers. Indeed, we obtain 100% accuracy in detecting attack-and-stop attacks.

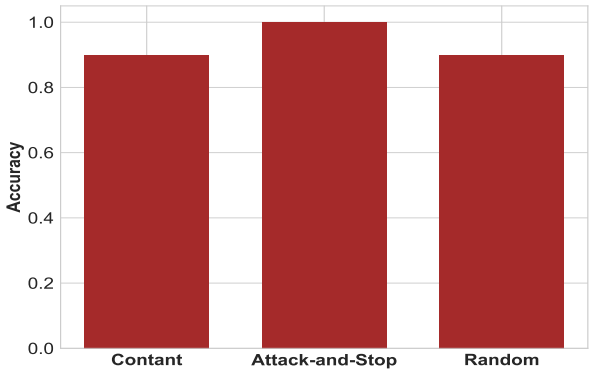


Fig. 15: Accuracy of detecting simulated attacks

VIII. Discussion

In this section, we discuss the technical aspects of our scheme. We also discuss the obtained results and give some recommendations.

Our scheme allows detecting passive mobile attackers based on their broadcast beacons. Since the time between two consecutive beacons is less than 1s, the feature extraction is quickly performed. Consequently, the attackers are detected fast. However, some attack behaviors could be similar to normal behaviors. For example, when two communicating vehicles keep the same speed for some time, the system could interpret this as a constant attack since the inter-distance between them remains unchanged. To avoid these false positives cases, vehicles should continue monitoring the suspicious vehicles for a certain period before reporting them to CA. On the other hand, in our scheme, the global ML model was build based on synthetic data. As previously mentioned using synthetic data can encompass all the envisioned scenarios, which are hard to cover through simulations or experiments. Generating synthetic data could also be guided by real traffic data to avoid implausible data.

From a security perspective, our scheme ensures that all FL-servers and selected FL clients (vehicles) are authenticated by the Certification Authority (CA). In addition, our scheme considers that all the communications links between FL clients and FL servers and between FL servers themselves are secure to protect gradients from any modification. On the other hand, the incentive is naturally ensured by our scheme since we are in a win-win situation .i.e vehicles aim to accurately detect who track their positions, they will be happy to participate in the scheme as FL client to enhance the detection accuracy. Our scheme is also privacy-preserving at different levels. Since federated learning is used, local private are not shared with edges nodes, which allows preserving location privacy of vehicles. Moreover, reporting potential attackers to CA is performed using their pseudonyms. Only CA can link pseudonyms to real identifiers of attackers.
Furthermore, our scheme is scalable and can manage the mobility of vehicles. The experiments show that our scheme can achieve high accuracy in a few rounds. Indeed, if five FL clients each round, accuracy can reach 95% in only 60 rounds. This number of rounds can easily be achieved using the synchronization between FL servers as described in section III. On the other hand, our self-labeling approach archives 80% accuracy, which is a good accuracy value that can be enhanced in our future works.

IX. Conclusion

This paper proposed a scheme for detecting passive mobile attacks in 5G Vehicular Edge Computing. This scheme leverages federated learning to enable secure and privacy-preserving collaborative learning for building an efficient global ML model to detect passive attackers. Moreover, by deploying FL servers at the edge, our scheme offers fast interaction with FL clients, which use semi-supervised learning to self-labeling data. Experiments have demonstrated that our solution can detect passive mobile attacks quickly and achieve higher detection accuracy in a short time. To the best of our knowledge, we are the first to propose a passive mobile attacks detection scheme for connected vehicles. As future work, we plan to perform some enhancements and carry out more simulations.

ACKNOWLEDGMENT

This work was supported by the 5G-INSIGHT bilateral project, (ANR-20-CE25-0015-16), funded by the Luxembourg National Research Fund (FNR), and by the French National Research Agency (ANR). This work is also supported by the 5G-MOBIX project. This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 825496. Content reflects only the authors' view and European Commission is not responsible for any use that may be made of the information it contains.

REFERENCES

[1] "Autopilot — Tesla," https://www.tesla.com/autopilot, accessed: 2020-09-02.

[2] L. Liu, C. Chen, Q. Pei, S. Maharjan, and Y. Zhang, "Vehicular edge computing and networking: A survey," *arXiv preprint arXiv:1908.06849*, 2019.

[3] J. A. Onieva, R. Rios, R. Roman, and J. Lopez, "Edge-assisted vehicular networks security," *IEEE Internet of Things Journal*, vol. 6, no. 5, pp. 8038–8045, 2019.

[4] D. Eckhoff and C. Sommer, "Driving for big data? privacy concerns in vehicular networking," *IEEE Security & Privacy*, vol. 12, no. 1, pp. 77–79, 2014.

[5] "Criminals use technology to track victims," https://www.timesdaily.com/archives/criminals-use-technology-to-track-victims, accessed: 2020-09-02.

[6] J2945/1, "On-board system requirements for V2V safety communications," *SAE Standards*, 2016.

[7] ETSI TR 103 415, "Intelligent Transport Systems (ITS); security; pre-standardization study on pseudonym change management," *ETSI standards*, 2018.

[8] B. Wiedersheim, Z. Ma, F. Kargl, and P. Papadimitratos, "Privacy in inter-vehicular networks: Why simple pseudonym change is not enough," in *2010 Seventh international conference on wireless on-demand network systems and services (WONS)*. IEEE, 2010, pp. 176–183.

[9] A. Boualouache, S.-M. Senouci, and S. Moussaoui, "A survey on pseudonym changing strategies for vehicular ad-hoc networks," *IEEE Communications Surveys & Tutorials*, vol. 20, no. 1, pp. 770–790, 2017.

[10] S. Lefevre, J. Petit, R. Bajcsy, C. Laugier, and F. Kargl, "Impact of v2x privacy strategies on intersection collision avoidance systems," in *Vehicular Networking Conference (VNC), 2013 IEEE*, Dec 2013, pp. 71–78.

[11] H. Chen, B. Guo, Z. Yu, and Q. Han, "Crowdtracking: Real-time vehicle tracking through mobile crowdsensing," *IEEE Internet of Things Journal*, vol. 6, no. 5, pp. 7570–7583, 2019.

[12] S. Gyawali, Y. Qian, and R. Q. Hu, "Machine learning and reputation based misbehavior detection in vehicular communication networks," *IEEE Transactions on Vehicular Technology*, 2020.

[13] C. H. Quevedo, A. M. Quevedo, G. A. Campos, R. L. Gomes, J. Celestino, and A. Serhrouchni, "An intelligent mechanism for sybil attacks detection in vanets," in *ICC 2020-2020 IEEE International Conference on Communications (ICC)*. IEEE, 2020, pp. 1–6.

[14] A. Le and C. Maple, "Shadows don't lie: n-sequence trajectory inspection for misbehaviour detection and classification in vanets," in *2019 IEEE 90th Vehicular Technology Conference (VTC2019-Fall)*. IEEE, 2019, pp. 1–6.

[15] H. Tan, Z. Gui, and I. Chung, "A secure and efficient certificateless authentication scheme with unsupervised anomaly detection in vanets," *IEEE Access*, vol. 6, pp. 74 260–74 276, 2018.

[16] K. M. A. Alheeti, A. Gruebler, and K. D. McDonald-Maier, "On the detection of grey hole and rushing attacks in self-driving vehicular networks," in *2015 7th Computer Science and Electronic Engineering Conference (CEEC)*. IEEE, 2015, pp. 231–236.

[17] M. Kim, I. Jang, S. Choo, J. Koo, and S. Pack, "Collaborative security attack detection in software-defined vehicular networks," in *2017 19th Asia-Pacific Network Operations and Management Symposium (APNOMS)*. IEEE, 2017, pp. 19–24.

[18] N. Negi, O. Jelassi, H. Chaouchi, and S. Clemençon, "Distributed online data anomaly detection for connected vehicles," in *2020 International Conference on Artificial Intelligence in Information and Communication (ICAIIC)*. IEEE, 2020, pp. 494–500.

[19] X. Li, Z. Hu, M. Xu, Y. Wang, and J. Ma, "Transfer learning based intrusion detection scheme for internet of vehicles," *Information Sciences*, 2020.

[20] T. Zhang and Q. Zhu, "Distributed privacy-preserving collaborative intrusion detection systems for vanets," *IEEE Transactions on Signal and Information Processing over Networks*, vol. 4, no. 1, pp. 148–161, 2018.

[21] A. König, R. Ackermann, M. Hollick, and R. Steinmetz, "Geographically secure routing for mobile ad hoc networks: A cross-layer based approach," 2007.

[22] M. Raya and J. Hubaux, "Securing vehicular ad hoc networks," *Journal of Computer Security*, vol. 15, no. 1, pp. 39–68, jan 2007.

[23] K. Emara, W. Woerndl, and J. H. Schlichter, "Vehicle tracking using vehicular network beacons," in *IEEE 14th International Symposium onA World of Wireless, Mobile and Multimedia Networks, WoWMoM 2013, Madrid, Spain*. IEEE, 2013, pp. 1–6.

[24] R. J. Fitzgerald, "Development of practical pda logic for multitarget tracking by microprocessor," in *American Control Conference, 1986*. IEEE, 1986, pp. 889–898.

[25] D. B. Reid, "An algorithm for tracking multiple targets," *Automatic Control, IEEE Transactions on*, vol. 24, no. 6, pp. 843–854, 1979.

[26] M. Feiri, J. Petit, and F. Kargl, "The case for announcing pseudonym changes," in *Proceeding of the 3rd GI/ITG KuVS Fachgespräch Inter-Vehicle Communication (FG-IVC 2015)*, 2015, pp. 31–33.

[27] J. Petit, D. Broekhuis, M. Feiri, and F. Kargl, "Connected vehicles: Surveillance threat and mitigation." Black Hat Europe, 11/2015 2015, pp. 1–12.

[28] L. Buttyán, T. Holczer, and I. Vajda, "On the effectiveness of changing pseudonyms to provide location privacy in vanets," in *Proceedings of the 4th European conference on Security and privacy in ad-hoc and sensor networks*, ser. ESAS'07. Berlin, Heidelberg: Springer-Verlag, 2007, pp. 129–141.

[29] R. W. van der Heijden, T. Lukaseder, and F. Kargl, "Veremi: A dataset for comparable evaluation of misbehavior detection in vanets," in *International Conference on Security and Privacy in Communication Systems*. Springer, 2018, pp. 318–337.

[30] H. Izakian, W. Pedrycz, and I. Jamal, "Fuzzy clustering of time series data using dynamic time warping distance," *Engineering Applications of Artificial Intelligence*, vol. 39, pp. 235–244, 2015.

[31] Z. Du, C. Wu, T. Yoshinaga, K.-L. A. Yau, Y. Ji, and J. Li, "Federated learning for vehicular internet of things: Recent advances and open issues," *IEEE Open Journal of the Computer Society*, vol. 1, pp. 45–61, 2020.

[32] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y Arcas, "Communication-efficient learning of deep networks from decentralized data," in *Artificial Intelligence and Statistics*, 2017, pp. 1273–1282.

[33] K. Bonawitz, H. Eichner, W. Grieskamp, D. Huba, A. Ingerman, V. Ivanov, C. Kiddon, J. Konečnỳ, S. Mazzocchi, H. B. McMahan *et al.*, "Towards federated learning at scale: System design," *arXiv preprint arXiv:1902.01046*, 2019.

[34] S. Samarakoon, M. Bennis, W. Saad, and M. Debbah, "Distributed federated learning for ultra-reliable low-latency vehicular communications," *IEEE Transactions on Communications*, vol. 68, no. 2, pp. 1146–1159, 2019.

[35] Y. Lu, X. Huang, Y. Dai, S. Maharjan, and Y. Zhang, "Differentially private asynchronous federated learning for mobile edge computing in urban informatics," *IEEE Transactions on Industrial Informatics*, vol. 16, no. 3, pp. 2134–2143, 2019.

[36] Y. Liu, J. James, J. Kang, D. Niyato, and S. Zhang, "Privacy-preserving traffic flow prediction: A federated learning approach," *IEEE Internet of Things Journal*, 2020.

[37] Y. Lu, X. Huang, K. Zhang, S. Maharjan, and Y. Zhang, "Blockchain empowered asynchronous federated learning for secure data sharing in internet of vehicles," *IEEE Transactions on Vehicular Technology*, vol. 69, no. 4, pp. 4298–4311, 2020.

[38] Y. Lu, X. Huang, Y. Dai, S. Maharjan, and Y. Zhang, "Federated learning for data privacy preservation in vehicular cyber-physical systems," *IEEE Network*, vol. 34, no. 3, pp. 50–56, 2020.

[39] D. Ye, R. Yu, M. Pan, and Z. Han, "Federated learning in vehicular edge computing: A selective model aggregation approach," *IEEE Access*, vol. 8, pp. 23 920–23 935, 2020.

[40] N. I. Mowla, N. H. Tran, I. Doh, and K. Chae, "Federated learning-based cognitive detection of jamming attack in flying ad-hoc network," *IEEE Access*, vol. 8, pp. 4338–4350, 2019.

[41] C. G. Cordero, E. Vasilomanolakis, A. Wainakh, M. Mühlhäuser, and

S. Nadjm-Tehrani, "On generating network traffic datasets with synthetic attacks for intrusion detection," *arXiv preprint arXiv:1905.00304*, 2019.
[42] V. Belenko, V. Krundyshev, and M. Kalinin, "Synthetic datasets generation for intrusion detection in vanet," in *Proceedings of the 11th International Conference on Security of Information and Networks*, 2018, pp. 1–6.
[43] 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, Jan 2011.
[44] SUMO, "Simulation of urban mobility," http://sumo.sourceforge.net/, 2019.