A Generative Adversarial Approach for Sybil Attacks Recognition for Vehicular Crowdsensing

Luis G. Jaimes *, Juan Calderon[†‡], Scott Shriver*, Antonio Hendricks[§], Javier Lozada[§], Sivasundaram Seenith[‡], Harish Chintakunta[§]
*Department of Computer Science, Florida Polytechnic University Lakeland, FL 33805
{ljaimes, sshriver2380}@floridapoly.edu
[§] Department of Electrical and Computer Engineering, Florida Polytechnic University
{ahendricks7304, jlozada7716, hchintakunta}@floridapoly.edu
[†]Department of Electronic Engineering, Universidad Santo Tomás, Colombia
juancalderon@usantotomas.edu.co
[‡]Computer Science & Engineering Department, Bethune Cookman University
{calderonj, sivasundarams}@cookman.edu

***Abstract*—Vehicular crowdsensing (VCS) is a subset of crowdsensing where data collection is outsourced to group vehicles. Here, an entity interested in collecting data from a set of Places of Sensing Interest (PsI), advertises a set of sensing tasks, and the associated rewards. Vehicles attracted by the offered rewards deviate from their ongoing trajectories to visit and collect from one or more PsI. In this win-to-win scenario, vehicles reach their final destination with the extra reward, and the entity obtains the desired samples. Unfortunately, the efficiency of VCS can be undermined by the Sybil attack, in which an attacker can benefit from the injection of false vehicle identities. In this paper, we present a case study and analyze the effects of such an attack. We also propose a defense mechanism based on generative adversarial neural networks (GANs). We discuss GANs' advantages, and drawbacks in the context of VCS, and new trends in GANs' training that make them suitable for VCS.**

Fig. 1: Vehicular crowdsensing System

I. Introduction

Vehicular Crowdsensing (VCS) is a new computation paradigm in which participants (vehicles) agree to use their onboard sensors to collect sensing samples as they go with their daily routines. A VCS system is composed of a cloud-based platform or crowdsoucer (data buyer), and a set of vehicles that collect data for the platform. The platform advertises a set of sensing tasks at some PsI and the corresponding rewards for completing each of these tasks. Attracted by the offered rewards some of the vehicles in the nearby area evaluate the convenience of visiting one or more PsIs.

Figure 1 shows a set of sensing tasks located in some regions of a city. Some are located in popular places such as highways and popular roads, while most of them are located in remote and isolated areas. Here, a vehicle requires a minimum effort to collect samples located in its own trajectory, in contrast to the effort required to collect samples from remote locations. Thus, the challenge consists in pursuing vehicles to deviate from their pre-planned trajectories to visit these PsIs. Incentive mechanism design approaches this problem from two different perspectives, namely auction-based [1], and platform-based mechanisms [2]. Auction-based approaches usually have the form of recurrent reverse auctions in which participants decide the value of their collected samples by submitting a bid price. On the other hand, in the platform-based approach, a participant's reward depends on its level of contribution to a sensing task versus other participants' contributions to the same task.

As autonomous vehicles become more prevalent in society, opportunities to use those vehicles to perform environmental analysis will present themselves. An anomalous vehicle trajectory could present an individual spoofing themselves as a legitimate vehicle in order to obtain financial incentives for performing a sensing task. Unfortunately, VCS systems are not immune to false identity or Sybil attacks [3] in which malicious agents may benefit from fabricating and using fake vehicles identities. The impact of these attacks on auction-based MCS has been widely studied from an incentive mechanism perspective [4], and from a mobile-location-based perspective [5]. However, unlike the aforementioned research, our focus is on the potential threats that Sybil attacks pose on platform-based VCS systems. To the best of our knowledge, this is the first work that analyses the effect of Sybil attacks on platform-based VCS systems.

Here, the discussion is driven by a case study [6] which allows to illustrate attacks to both the incentive mechanism, and the mobility component of the system. We present a deep learning approach to minimize the effect of Sybil attacks on VCS based on adversarial generative networks. Here, we

cast the problem of recognizing fake vehicles identities to the problem of vehicles' trajectories anomaly detection. Thus, the main goal of this approach is to make the system immune to Sybil attacks by allowing the platform and honest participants to learn how to recognize and ignore fake vehicles.

II. Literature Review

A pioneer work on Sybil attacks on vehicular networks is the work of Gang Wang [3] who demonstrated that Sybil attacks can be performed to compromise the integrity of crowdsourced map applications. By using multiple instances of an emulator of a mobile device operating system with the app Waze installed, they were able to control those emulated devices GPS and simulate device movements to give the impression of being an authentic vehicle.

Zhang [7] proposed the use of Long Short-Term Memory (LSTM) to extract features from packet data of electric vehicles in order to differentiate legitimate vehicles from Sybil attacks. In a VANET architecture, vehicles broadcast CAM packets that contain information such as location, elevation, and time continuously. By measuring the strength of the signal, or RSSI, of these CAM packets, other vehicles or nodes can determine if the disparity between broadcasted location and the RSSI are realistic or not.

James Yu [8] proposed a deep learning approach to this anomaly detection called BRAE, or Bayesian Recurrent Autoencoder. BRAE represents trajectories as random multivariate data in a latent trajectory space and can determine the probability of a trajectory being authentic or not.

Another approach to detecting anomalous trajectories is the GM-VSAE [9]. A Gaussian Mixture Variational Sequence Autoencoder, model proposed by Yiding Liu. This model proposes a method of identifying anomalous trajectories in real time online. This model comprises a route inference network made of an RNN as well as a generative network designed to assist in the real-time anomaly detection.

Another approach to anomalous trajectory detection is proposed by Wang [10]. They argue that many approaches to anomaly detection run into two problems: modeling normal routes is very difficult due to the wide variety of possible routes, and training is incredibly difficult due to the sparsity of authentic route data for training. They propose using GCM, or Geospatial Constraint Modeling, to overcome these issues. A unique aspect to this GCM approach is that rather than looking exclusively at the trajectory as a linear sequence, this model looks for a relationship between the pairing of the source and destination points against the midway points in the trajectory. The source and destination points are considered the constraints for the trajectory, and are used to consider the probability of all points in between the constraints. [10]

Another approach called TrajGAN [11] aims to use GANs combined with a Partially Observable Markov Decision Process to generate trajectories. It works by breaking a series of continuous latitude and longitude pairs into link sequences, where at each link sequence the vehicle can decide to move straight, left, right, or stop the vehicle. A value estimator is used to calculate the return or accuracy of the possible moves, with those values then used by a policy generator to make a decision on how to get closer to the end of the trajectory. [11]

.

III. System Model and Adversarial Model

A. Vehicular crowdsensing system

Here, we present the most relevant elements of our case study which the reader can review in detail in [6]. Our VCS system consists of a set of crowdsourcers $C = \{c_1, c_2, \ldots, C_F\}$, and a set of participants $V = \{v_1, v_2, \ldots, v_M\}$ as shown Figure 2. Each crowdsourcer c_j advertises a sensing task and the corresponding reward R_j. We assume participants are rational players who respond to the crowdsourcer offers if they obtain a positive utility.

The utility of a participant sensing for crowdsourcer j is define by

$$u_i^j = \frac{t_i^j}{\sum_{i \in W_j} t_i^j} R_j - k_i^j t_i^j \tag{1}$$

where W_j is the set of participants working on task j at the same time than particiant i, with cardinality of w_j, k_i^j is the cost per unit of sensing data in which i incurs for working for crowdsourcer j and t_i^j is the number of samples (sensing plan) i collects from j.

The utility of crowdsourcer j is proportional to the samples provided by their workers (vehicles) minus the reward paid for the data, and is define as $\bar{u}_j = \lambda_j log\big(1 + \sum_{i \in W_j} log(1 + t_i^j)\big) - R_j$.

A participant decides to deviate from its ongoing trajectory to visit and collect from a crowdsourcer j if it can obtain a positive utility. However, it is impossible to predict in advance that utility, given the uncertain behavior of other participants, namely once participant i arrives at crowdsourcer j it may have to share the reward R_j with none, one or several participants who may be already collecting there. Equation 2 corresponds to expecting utility of participant i visiting any crowdsourcer j and finding 0,1,2,..participants already collecting there. Here, A is the set participants with any chance to reach reach j at the same time that i. This, dynamic is naturally model by the Poisson binomial distribution [12]

$$EU_i^j = \sum_{A \subset V, i \in A} \prod_{j \in A} P_j \times \prod_{j \notin A} (1 - P_j) \times u_i | A \tag{2}$$

Another important factor to take into account is the optimal amount of data a participant v_i should collect (sensing plan t_i^j) when arriving at crowdsourcer c_j, and finds $k_{w_j} - 1$ other participants already there. Here, we use Algorithm 1 to compute Nash Equilibrium (NE) sensing plan for all the k_{w_j} participants working at the same time for c_j

Thus, all the participants working for crowdsourcer c_j submit their costs to c_j, c_j runs Algorithm 1 and returns to each participant the corresponded sensing plan t. Unfortunately, it is impossible for participant v_i to compute in advance the sensing plan of working for crowdsourcer c_j (t_i^j) without reaching physically c_j. This, because v_i doesn't know in advance how

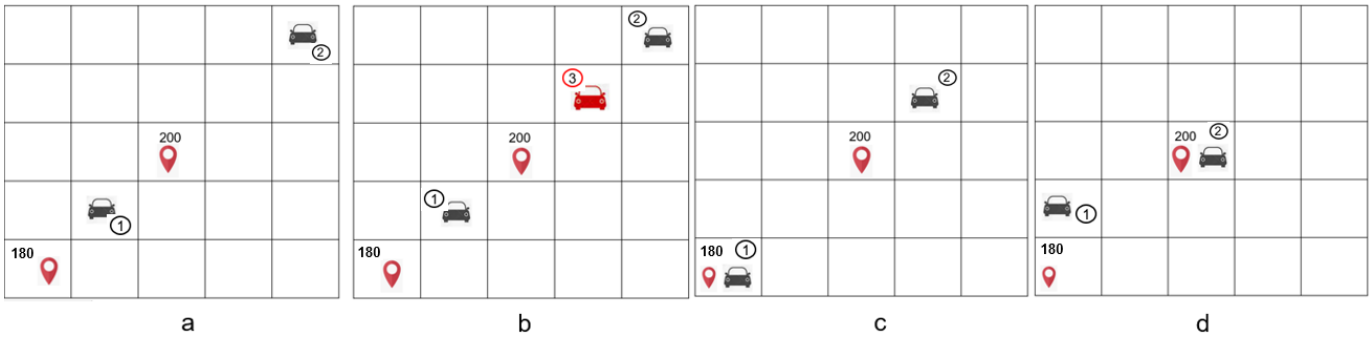

Fig. 2: Example of a Sybil attack

Algorithm 1: Computation of the Nash equilibrium Sensing Plans for Participants in Crowdsourcer j

Sort the set of contributors W_j ($|W_j| = w_j$) in crowdsourcer j according to their costs, $k_1^j \leq k_2^j \leq \cdots \leq k_{w_j}^j$;
$H \leftarrow \{1, 2\}, i \leftarrow 3$;
while $i \leq w_j$ and $k_i^j \leq \frac{k_i^j + \sum_{l \in H} k_l^j}{|H|}$ **do**
  $H \leftarrow H \cup \{i\}, i \leftarrow i + 1$;
end while
for all $i \in W_j$ **do**
  if $i \in H$ **then**
    $(t_i^j)^* = \frac{(|H|-1)R_j}{\sum_{l \in H} k_l^j}\left(1 - \frac{(|H|-1)k_i^j}{\sum_{l \in H} k_l^j}\right)$;
  else
    $(t_i^j)^* = 0$;
  end if
end for
return $((t_1^j)^*, (t_2^j)^*, \ldots, (t_{w_j}^j)^*)$;

many other participants it will find when arriving to c_j . Thus, v_i compute the expected sensing plan ESP_i^j to evaluate the convenience of deviating from its pre-planned trajectory to visit c_j. This probabilistic computation has form of Equation 3 and it is naturally model by the Poisson binomial distribution.

$$ESP_i^j = \sum_{A \subset V, i \in A} \prod_{j \in A} P_j \times \prod_{j \notin A} (1 - P_j) \times t_i | A \qquad (3)$$

Figure 3 shows the flow of the decision making process of

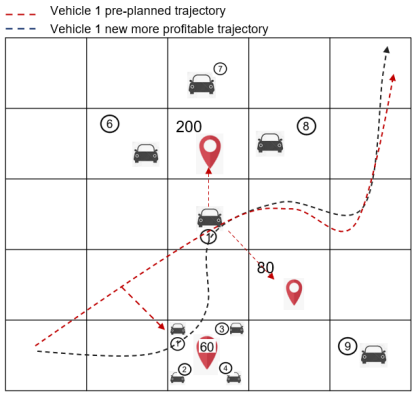

Fig. 3: Evaluating next location to move

a vehicle (vehicle 1) whose move from source to destination. Here, the red dash line corresponds to vehicle 1's pre-planned trajectory, while the blue one corresponds to the new trajectory that results from vehicle 1's vising some crowdsourcers. The first deviation occurs when it decides to visit a crowdsoucer offering a reward of 60. There, vehicles 1 finds that vehicles 3, 4, 5 are already there. Thus, vehicles 1, 3, 4, 5 submit their cost to the corresponding crowdsourcer who computes their sensing plans using Algorithm 1. Vehicles 1, 3, 4, 5 can now use their sensing plans values to compute their actual utilities. Vehicles 1 continues its journey and when it reaches the center of the grid, and it has to decide whether to head to its final destination or deviating again to visit either the crowdsourcer that offers 200 or the one offering 80. Here, it compares EU_1^{200} and EU_1^{80} and chooses to visit the crowdsourcer yielding the highest expected utility. However, before heading there, vehicle 1 computes the expected sensing plan, namely the expected optimal number of units of sensing data it needs to collect to maximize its utility. We assign a constrain in terms of the number of sensing samples a vehicle can collect during its entire journey, namely we assign to each vehicle a sensing capacity $B = \{b_1, b_2, \ldots, b_M\}$ to avoid any unrealistic infinite looping collection. Therefore, once a vehicle fills out its sensing capacity it just heads to its final destination.

B. Threat Models

Threats to Incentive: We assume that users are selfish but rational. Hence it is possible that user i maximizes its utility by reporting a false cost value, and false sensing capacity.

Sybil Attack: Based on our system model, a user could conduct Sybil attack by creating multiple fictitious identities around a target crowdsourcer. Thus, discouraging honest participants to visit these places, and thus downgrading the quality of the outcome.

1) Example 1 - Threat to the incentive mechanism : Here, a vehicle can fake its own cost to improve its utility or for jeopardizing the system. Figure 2 shows vehicles v_1, v_2, v_3 in the cell located at the center of the bottom row. Let's assume the sensing costs of $k_1 = 1, k_2 = 2, k_3 = 4$ respectively. Thus, each vehicles uses Algorithm 1 to compute its sensing time resulting in $t_1 = \frac{20}{0}, t_2 = \frac{10}{0}, t_3 = 0$. Thus, v_3 anticipating this unfavorable outcome decides to jeopardize the efficiency

of the crowdsourcer by injecting two fake vehicles with costs $k_4 = 0.3, k_5 = 0.5$. This move will kicked out all the reeal participants v_1, and v_2 resulting in sensing plans $t_1 = 0, t_2 = 0, t_3 = 0, t_4 = 7.81, t_5 = 4.37$. Thus, the platform will never receive these samples.

2) Example 2 - Sybil attack : Figure 2a shows an scenario which includes two participants (black vehicles) and two sensing task or crowdsourcers (read marks) with their respective offered rewards (200, and 185). To simplify the example, let's assume vehicles move one step at the time with eight degrees of freedom in a synchronous fashion. Figure 1a shows that at time t vehicles 1 and 2 are evaluating visiting and collecting from the crowdsourcers offering rewards 200 and 185 respectively. At time $t + 1$ vehicle 1 naturally will visit and collect from the nearby sensing task offering a reward of 200. Vehicles 2 which is a malicious vehicle wants to stop vehicle 1 from getting all the reward (200). Figure 1b shows vehicle 2 creating fake vehicle 3 (red) at time t. Vehicle 1 then uses Equation 2 to compute EU_1^{200}, and EU_1^{815} respectively. Let's simplify the computation by assuming equal sensing cost. Then, Equation 1 simplifies to $\frac{R_j}{w_j^2}$ where w_j is the number of participants working for crowdsourcer j. Thus, $EU_1^{200} = u(1)(P_1^{200} * (1 - P_2^{200})) + u(2)(P_1 * P_2)$ = $\frac{200}{1}(1 * \frac{7}{8} + \frac{200}{4}(1 * \frac{1}{8}) = 181.25$, and $EU_1^{185} = u(1)P_1 = \frac{185}{1} * 1 = 185$. Where $u(1)$,and $u(2)$ are the utilities that result from finding one and two participants included i when i arrives to crowdsourcer j. Figure 1c shows vehicle 1's new position at time $t + 1$. Here, vehicle 1 is visiting and collecting from the crowdsourcer yielding the highest utility (185). Figure 1d shows now vehicle 2 landing with no competition and collecting from crowdsourcer with $R = 200$. Here, vehicle 2 was able to deceive vehicle 1 by injecting fake vehicle 3 in vehicle 1's expected utility computation.

Here, it is reasonable to assume the platform can track in real-time participants using their GPS signals, and then make this information available to every participant. In this scenario, fake vehicles can not just appear from out of the blue, their trajectories have to be carefully fabricated to have some chance to deceive the platform. Thus, we propose a defense mechanism based on the idea of recognizing fake participants based on their trajectories. The problem of building such a classifier is the lack of training data, namely the availability of fake trajectories that allows training such a classifier. Thus, generative models seem to be a suitable framework to address this problem.

C. Casting Sybil Recognition to Trajectory Anomaly Detection

Let $V = \{v_1, v_2, \ldots, v_M\}$ be the set of M participant vehicles, and $S = \{s_1, s_1, \ldots, s_M\}$ and $D = \{d_1, d_2, \ldots, d_M\}$ their starting and final destination locations. In addition, we consider a time-stamp associated with each of its GPS coordinates. We model the set of M vehicles trajectories as a set of multi-variable (longitude, latitude, time-stamp) time series of different lengths. Thus, our approach consists in identify a vehicle as a potential Sybil agent if the trajectory generated as it travels from source to destination is out of the distribution of trajectories generated by normal driving behaviors under the actual traffic conditions. Formally speaking, consider T is the set of all the possible trajectories available to all the vehicles V under any driving condition, and $\hat{T}$ the set of trajectories generated by elements of V when following a normal travel behavior under the current traffic conditions. Thus, we say that vehicle v_i with trajectory T_i is a Sybil agent if $P(T_i|\hat{T})$ $<$ Th, where the Th is a learned parameter. In the absence of fake or adversarial trajectories to build a trajectory classifier (supervised), we found that the semi-supervised Generative Adversarial Model (GAN) is a good framework to tackle this task.

D. The Generative Adversarial Model (GAN)

The idea of GAN was motivated by the concept of Nash Equilibrium (NE) in game theory. A GAN has two components, the generator (G) whose job is to generate samples from a distribution and make them pass as real samples, and the discriminator (D) which checks whether the input samples are coming from the distribution of the real or from the generation of the generator. The discriminator and the generator then form a min-max game, where both sides continuously optimize themselves in the training stage and are ultimately supposed to archive the NE. The training of the D to maximize the probability of assigning the correct trajectories to training examples and samples from the generator. This could be determined if a sample $x \in X$ is from the real data set $x \sim p_{data}(x)$ or generated from the generator G , $z \in G, z \sim p_z$. The objective function for GAN if formulated as follows.

$$\min_G \max_D V(D, G) = E_{x \sim p_{data}(x)}[logD(x)] +$$

$$E_{z \sim p_z}[log(1 - D(G(z)))]$$

Discriminator D should enforce the output $D(x)$ close to 1, while enforcing the output of$D(G(z))$ close to 0. The loss function can be considered as a cross-entropy.

However, the cross-entropy loss function used in the GAN can easily lead to the disappearance of gradient, which makes the network difficult to train. Secondly in addition mode collapse might be generated when generators only learn local features of the trajectories. In order to overcome these two problems, we use particle swarm optimization which explain next. **Particle swarm optimization (PSO)** The global optimum solution of the swarm particles defines as initial fitness value denoted by x. The swarm consists of n-particles travelling into n-dimensional search space with epoch t. During each epoch, p particle produce a unique position vector x. The cost function is also calculated by each particle p which is considered as the local best fitness (p_k^{best}) to fined the best fitness called global best fitness (g_k^{best}) of the swarm. The PSO is formulated as follows

$$v_{k+1}^t = v_k^t + c_p(p_k^{best} - x_k^t) + c_g(g_k^{best} - x_k^t)$$

$$x_{k+1}^t = x_k^t + v_{k+1}^t$$

where x_k^t and v_k^t are the position vector and velocity vector of epoch t, The PSO is used to optimize the parameters of the generator network, where each particle represents a generator. The length of the particle is the number of parameters that need to be learned in the generator network. In our simulations, we use Adam's optimization algorithm to update network weights iterative based on training data. Algorithm 2 sketches the main components of our proposed PSO-based GAN.

Algorithm 2: PSO-Based GAN

Input: Particle number P_n, the minibatch size m, the discriminator's updating steps per iteration k, Adam's hyper parameters α, β_1, β_2
for all training iterations **do**
  for all k steps **do**
    Sample of minibatch of m examples$x^1, , x^m$) from data generating distribution $p_{data}(x)$
    Sample of minibatch of m noise samples $z^1, , z^m$) from p_z

$$g_{\theta d} \leftarrow \nabla_{\theta d}[\frac{1}{m}\sum_{1}^{m}[logD(x^i) + log(1 - D(G(z^i)))]$$

$$\theta d \leftarrow Adam(g_{\theta d}, \theta d, \alpha, \beta_1, \beta_2)$$

$$g_{\theta g} \leftarrow \nabla_{\theta g}[\frac{1}{m}\sum_{1}^{m}[log(1 - D(G(z^i)))]$$

$$\theta g \leftarrow Adam(g_{\theta g}, \theta g, \alpha, \beta_1, \beta_2)$$

  end for
  Initialize x_k swarms velocity vector v_k, local (p_k^{best}) and global (g_k^{best}), evaluate fitness function and update (p_k^{best}), (g_k^{best})
  for all j **do**
    Sample of minibatch of m noise samples
    Evaluate P_j
    $P_{gbest}, P_{ibest} \leftarrow P_j$
  end forupdate x_k swarms positions velocity vector v_k
  for all n **do**
    Update particle x_n, v_n
  end for
end for

E. Bidirectional GAN (BiGAN) for Anomaly trajectory detection

In this work, we use IGMM-GAN [13] an extension of the BiGAN model [14] for anomaly detection of multi-modal trajectory data. BiGAN is a GAN extension that allows the generator not only to transform data from a distribution z into fake generated trajectory $G(z)$, but also introduce an encoder E that allows generating a latent space $E(x)$ from a real sample x. Thus, the discriminator D not only have to learn to distinguish between $G(z)$, and x, but also now between (tuples(x, $E(X)$)) and ($G(z)$,z)). Thus, once learned the parameters of the distribution (mean, and standard deviation) in the latent space, we can easily compute an anomaly score by measuring the distance of a new encoded test sample in the latent space to the learned cluster (distribution of the latent space). Finally, the IGMM extends the BiGAN framework by introducing an Infinite Gaussian Mixture Model which allows finding the number of clusters in the latent space. This addition allows to compute anomaly scores in multi-modal time series such trajectories with components in space (GPS coordinates), and time (timestamp). These scores are proportional to the Mahalanobis distance between the clusters in latent space and any new encoded trajectory in the test set.

IV. Performance Evaluation

This section presents the preliminary results of our GAN-based Sybil recognition system.

A. Experimental Setup

TABLE I: Simulation Parameteres

Parameters		**Parameters (cont.)**	
Target Area	5200 x 5200 mt	Distance Capacity	7
Cell Size	100 x 100 mt	Crowdsourcer Radius	30
Crowdsourcers	20	Memory Size	2
Crowdsoucer location	random	Gate Buffer Interval	30
Reward Distribution		Simulation count	3
μ	1000		
σ	5		
Participant Capacity		Autonomous Vehicles	
μ	$10 - 90$	*Trajectory source*	random
σ	5	*Trajectory destination*	(0, 0)
		Num vehicles	150

1) trajectory generation: Table 1 shows the system parameters for trajectory generation. We generate 150 trajectories by running the Approximated Trajectory Nash Equilibrium (ATNE) Vehicular crowdsensing system described in [6]. ATNE uses the simulation of Urban Mobility SUMO [15] capabilities to simulate realistic vehicular displacement, Open Street Maps (OSM) [16] for mapping, and velocity information per street from Uber Movement speed dataset [17]. The simulation takes place in a sub-region of 5200x5200 meters from the downtown of the city of London.

Here, each trajectory includes the vehicle ID, latitude, longitude, time, velocity, and acceleration. In this scenario, trajectories center around vehicles deviating from their ongoing trajectory to visit and collect from places of sensing interest, namely these trajectories resemble real driving conditions in the context of vehicular crowdsensing.

We train the discriminator using trajectories starting from random places on the map. Here, the vehicles follow the ATNE protocol for data collection and head to their final destination which corresponds to location coordinate (0, 0)

For the purpose of this experiment we use a batch size of 50, and 40,000 epochs.

B. Experiment

The goal of the first experiment is to compute the ability of the GAN model to discriminate between real and fake participants. Here, we use the 150 trajectories from participants of a vehicular crowdsensing system (VCS) whose mobility patterns are influenced by ATNE [6] incentive mechanism. Here, the set of trajectories is split in an 80/20 partition for training and validation. In addition, eight participants whose mobility patterns follow a random deviation from the original trajectories as described in [6] are used as a held out class. Here, we set one trajectory at the time from the held-out set as the anomalous class, and train for 40,000 epochs. We repeat

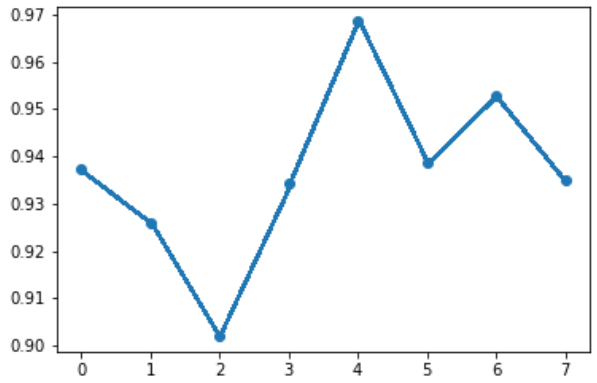

Fig. 4: ROC AUC scores of IGMM-GAN using SUMO data, x-axis: trajectories in the held-out set, y-axis: AUC-ROC score

this process for each trajectory in the held-out set. Figure 5, shows the Area Under the Curve of Receiver Operator Characteristic AUC-ROC scores for each trajectory in the held-out set. A higher score means the discriminator is less likely to classify an authentic trajectory as fake, and vice versa. The x-axis represents 8 test cases the discriminator went through, while the y-axis represents the AUC-ROC score. The results span the range of 90 to 97 percent.

Figure 6 shows the generator and discriminator loss by epoch and the accuracy of the discriminator for trajectory number 2. This figure is consistent with the low value of the AUC-ROC score for that trajectory

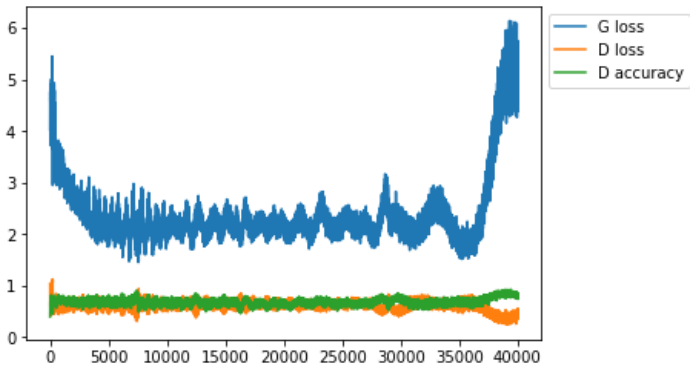

Fig. 5: Generator, and discriminator Loss. Accuracy of the discriminator for trajectory number 2 in held out class, x-axis: number of epochs, y-axis: loss, and accuracy

V. Conclusion

In this paper, we present a framework for Sybil attack detection in the context of platform-based vehicular crowdsensing systems. The proposed approach is based on vehicles' trajectory anomaly detection. Here, the discussion is driven by a particular vehicular crowdsensing case study. We present examples of attacks to both the incentive mechanism and the mobility model. In addition, we present the components of the framework which is based on generative adversarial models. Finally, we present preliminary results of our work.

Acknowledgment

This material is based upon work primarily supported by the National Science Foundation (NSF) under NSF Award Number 1739409. Any opinions, findings and conclusions, or recommendations expressed in this material are those of the author(s), and do not necessarily reflect those of the NSF.

References

[1] Luis G Jaimes, Idalides J Vergara-Laurens, and Andrew Raij. A survey of incentive techniques for mobile crowd sensing. *IEEE Internet of Things Journal*, 2(5):370–380, 2015.

[2] Dejun Yang, Guoliang Xue, Xi Fang, and Jian Tang. Crowdsourcing to smartphones: Incentive mechanism design for mobile phone sensing. In *Proceedings of the 18th annual international conference on Mobile computing and networking*, pages 173–184, 2012.

[3] Gang Wang, Bolun Wang, Tianyi Wang, Ana Nika, Haitao Zheng, and Ben Y Zhao. Ghost riders: Sybil attacks on crowdsourced mobile mapping services. *IEEE/ACM transactions on networking*, 26(3):1123–1136, 2018.

[4] Jian Lin, Ming Li, Dejun Yang, Guoliang Xue, and Jian Tang. Sybil-proof incentive mechanisms for crowdsensing. In *IEEE INFOCOM 2017-IEEE Conference on Computer Communications*, pages 1–9. IEEE, 2017.

[5] Mohamed Baza, Mahmoud Nabil, Mohamed Mohamed Elsalih Abdelsalam Mahmoud, Niclas Bewermeier, Kemal Fidan, Waleed Alasmary, and Mohamed Abdallah. Detecting sybil attacks using proofs of work and location in vanets. *IEEE Transactions on Dependable and Secure Computing*, 2020.

[6] Alireza Chakeri, Xin Wang, Quentin Goss, M Ilhan Akbas, and Luis G Jaimes. A platform-based incentive mechanism for autonomous vehicle crowdsensing. *IEEE Open Journal of Intelligent Transportation Systems*, 2:13–23, 2021.

[7] Yi-Ying Zhang, Jing Shang, Xi Chen, and Kun Liang. A self-learning detection method of sybil attack based on lstm for electric vehicles. *Energies*, 13(6):1382, 2020.

[8] JQ James. Sybil attack identification for crowdsourced navigation: A self-supervised deep learning approach. *IEEE Transactions on Intelligent Transportation Systems*, 2020.

[9] Yiding Liu, Kaiqi Zhao, Gao Cong, and Zhifeng Bao. Online anomalous trajectory detection with deep generative sequence modeling. In *2020 IEEE 36th International Conference on Data Engineering (ICDE)*, pages 949–960. IEEE, 2020.

[10] Haiquan Wang, Jiachen Feng, Leilei Sun, Kaiqiang An, Guoping Liu, Xiang Wen, Runbo Hu, and Hua Chai. Abnormal trajectory detection based on geospatial consistent modeling. *IEEE Access*, 8:184633–184643, 2020.

[11] Seongjin Choi, Jiwon Kim, and Hwasoo Yeo. Trajgail: Generating urban vehicle trajectories using generative adversarial imitation learning. *Transportation Research Part C: Emerging Technologies*, 128:103091, 2021.

[12] Yili Hong. On computing the distribution function for the poisson binomial distribution. *Computational Statistics & Data Analysis*, 59:41–51, 2013.

[13] Kathryn Gray, Daniel Smolyak, Sarkhan Badirli, and George Mohler. Coupled igmm-gans for deep multimodal anomaly detection in human mobility data. *arXiv preprint arXiv:1809.02728*, 2018.

[14] Jeff Donahue, Philipp Krähenbühl, and Trevor Darrell. Adversarial feature learning. *arXiv preprint arXiv:1605.09782*, 2016.

[15] Daniel Krajzewicz, Jakob Erdmann, Michael Behrisch, and Laura Bieker. Recent development and applications of sumo-simulation of urban mobility. *International journal on advances in systems and measurements*, 5(3&4), 2012.

[16] Ming Wang, Qingquan Li, Qingwu Hu, and Meng Zhou. Quality analysis of open street map data. *International archives of the photogrammetry, remote sensing and spatial information sciences*, 2:W1, 2013.

[17] Data retrieved from Uber Movement. (c) 2020 uber technologies., https://movement.uber.com. 1, 2,3.