Towards Synthetic Data Generation of VANET Attacks for Efficient Testing

Thomas Rosenstatter
RISE Research Institutes of Sweden
Gothenburg, Sweden
thomas.rosenstatter@ri.se

Kateryna Melnyk
RISE Research Institutes of Sweden
Lund, Sweden
kateryna.melnyk@ri.se

***Abstract*—Vehicle-to-Vehicle communication can improve traffic safety and efficiency. This technology, however, increases the attack surface, making new attacks possible. To cope with these threats, researchers have made a great effort to identify and explore the potential of cyberattacks and also proposed various intrusion or misbehaviour detection systems, in particular machine learning-based solutions. Simulations have become essential to design and evaluate such detection systems as there are no real publicly available Vehicular Ad-Hoc Network (VANET) datasets containing a variety of attacks. The drawback is that simulations require a significant amount of computational resources and time for configuration.**

In this paper, we present an attack simulation and generation framework that allows training the attack generator with either simulated or real VANET attacks. We outline the structure of our proposed framework and describe the setup of a standard-compliant attack simulator that generates valid standardised CAM and DENM messages specified by ETSI in the Cooperative Intelligent Transport Systems (C-ITS) standards. Based on the introduced framework, we demonstrate the feasibility of using deep learning for the generation of VANET attacks, which ultimately allows us to test and verify prototypes without running resource-demanding simulations.

***Index Terms*—VANET, cybersecurity, generative adversarial networks**

I. Introduction

In the last decade, vehicular systems have gained much attention in regard to cybersecurity, mainly due to the increasing use of external communication and the advancement in automated and autonomous driving. Vehicle-to-Vehicle (V2V) communication is one example of external communication which is seen as a key enabler for improving traffic safety as vehicles blocked by objects (out-of-sight) can be detected. V2V communication also increases traffic efficiency by introducing new functionalities like platooning and cooperative intersections [1].

Day one applications for cooperative driving are defined in the ETSI C-ITS standards, which include the communication stack based on EU ITS-G5, and the two types of C-ITS messages Cooperative Awareness Messages (CAMs) and Decentralised Environmental Notification Messages (DENMs). CAMs are heart-beat broadcast messages consisting of basic information such as the speed of a vehicle while DENMs are warning messages broadcast when a specific event such as an accident is detected.

Motivation. These advancements in Vehicular Ad-Hoc Networks (VANETs), however, also pose a higher risk to be targeted by an attacker (e.g., [2]) whose aim is to disrupt traffic or even cause accidents. To cope with this expanding threat landscape, it is needed to develop techniques for detecting attacks in VANETs. Research in various fields has been adopting machine learning methods for the development of Intrusion Detection Systems (IDSs), and also for the application in VANETs [3]. Designing machine learning-based IDSs requires an understanding of the type of communication and traffic data. The limited access to such data may cause current solutions to only focus on a small set of possible attacks or to rely on simulations [4]. Though using simulations in VANETs has the advantage to simulate corner cases and other rare attacks, many of these simulation frameworks are not open source or require a commercial license.

Contributions. We aim to address these challenges through our proposed framework named *CyberSAGE*. We extend the *ezCar2x* [5] framework, which is a VANET simulation framework developed and maintained by Fraunhofer IKS. It provides a detailed implementation of the ETSI ITS-G5 stack including CAM and DENM messages. The detailed implementation in ezCar2x enables us to implement VANET misbehaviour attacks that are ETSI standard-compliant and thus could be used for efficient testing of real modules. Due to the lack of real-world VANET attack traffic [4], we use the attack simulator for data collection and then train an attack generator based on generative adversarial networks (GANs). Such an attack generator has the advantage of requiring fewer computational resources once trained. It also allows us to generate new attack traffic that shares similar characteristics from the original dataset for a larger and more diverse dataset. More specifically, we (i) design a framework for the simulation and generation of synthetic data, (ii) demonstrate the feasibility of using GANs to generate VANET attacks, and (iii) provide an evaluation of the preliminary results.

II. Related Work

In this section, we describe existing work in VANET simulation with a focus on attack behaviour, the application

Both authors contributed equally to this work. This work was partially supported by the Swedish Innovation Agency Vinnova through the project ICV-Safe (2019-03418) and the FFI project DIFFUSE (2021-05038).

T. Rosenstatter, K. Melnyk, "Towards Synthetic Data Generation of VANET Attacks for Efficient Testing," *accepted in 2023 IEEE Intelligent Vehicles Symposium (IV2023)*, IEEE 2023.

of deep generative models to VANET attack data, and the advancements in generating time-series data.

A. VANET Attack Simulators

Attacks can be categorised according to the STRIDE threats, i.e., *spoofing*, *tampering*, *repudiation*, *information disclosure*, *denial of service* and *elevation of privilege*. Due to the nature of VANETs, many attacks focus on so-called misbehaviour attacks aiming to disrupt or re-route traffic by sending incorrect information (mainly *spoofing*). Other types of attacks focus on dropping messages (mainly *denial of service, DoS*) when they are routed between nodes (e.g., black hole and grey hole attacks) or re-transmit recorded traffic (e.g., replay attack) [2].

Kamel et al. [6] provide a comprehensive attack simulation framework focusing on misbehaviour in VANETs which also includes modules for local and global detection, such as speed consistency, range plausibility and position plausibility checks. One limitation of this framework is that it does not include ITS-G5 compliance up to the application layer, i.e., compliant implementations of CAMs and DENMs.

Iqbal et al. [7] focus on the generation of a learning dataset for VANET attacks. The authors use the Eclipse MOSAIC simulator [8] for generating the dataset, which gives other researchers the possibility to create their own scenarios, yet it also requires more computational resources for running the simulation. The attacks available in the provided dataset contain replay and bogus information attacks.

Belenko et al. [9] propose a simulator for generating VANET datasets by using the ns-3 network simulator [10] and further log routing tables and the distributed packets. The paper focuses on the generation of routing-based attacks like DoS, black hole, and grey hole attacks. The user can choose from 10 different pre-defined mobility models, e.g., city traffic in Paris. The authors, however, do not provide any details on the standards they followed.

B. VANET Attack Generation and time series signals

After the introduction of GANs in 2014 by Goodfellow et al. [11], they have been applied in various domains such as computer vision, natural language processing, and time series. The following part reviews previous work that employs GANs for generating VANET attacks, together with the latest developments on continuous-valued sequence generation.

Generating VANET attacks is challenging due to the variety of attacks and the difficulty to represent the feature space and target space in a way (as numerical vectors) that the neural network can handle. One method that utilises a GAN in a traffic scenario is introduced by Seo et al. [12]. Yet, the focus of their work is not the generation of synthetic data, but the development of an IDS based on GANs for detecting unknown in-vehicle attacks in CAN data. Shu et al. [13] present a collaborative misbehaviour detection system based on software-defined networking (SDN). The proposed approach uses GANs to enable multiple distributed SDN controllers to jointly train a machine learning model for the entire network. Evidently, not many works have been exploring the generation of VANET attacks. Most methods have focused on building IDSs, whereas we are aiming at investigating the use of deep generative models for generating synthetic data of VANET attacks.

Deep generation models for continuous-valued sequences such as time series have been drawing a great amount of attention in a variety of applications. The closest to our work is the generation of in-vehicle signals for testing low-level vehicle control software. Parthasarathy et al. [14] applied the Variational Autoencoder-Generative Adversarial Network (VAE/GAN) architecture to learn an unlabelled dataset of recorded in-vehicle signals and use it for the generation of synthetic input stimulus. To capture characteristics that are interesting for the test and customise stimulus generation, they introduced a template and metric-based linear interpolation algorithm. Parthasarathy et al. [15] improve on the work above by introducing a simpler way of using templates to set explicit objectives for domain adaptation. Moreover, they utilise unsupervised disentanglement representation to enrich the generation of the test stimulus. In [16], Zec et al. train a recurrent GAN to generate a time series of automotive perception sensors for simulation-based verification. In [17] and [18], the authors have applied convolutional GANs for sequence generation. Both works use a mixture density network and Wasserstein-1, while the latter additionally performs k-means clustering to evaluate the effectiveness of the generated synthetic time series datasets. Another method is TimeGAN [19] that generates realistic time series data from a leaned embedding space which in turn is optimised with both supervised and adversarial objectives.

III. The CyberSAGE Framework

In this paper, we propose a Cybersecurity Simulation and Attack Generation (CyberSAGE) framework. As illustrated in Figure 1, CyberSAGE provides an environment for attack simulation and generation to allow early verification and validation of VANET prototypes including machine learning-based solutions and more efficient testing to verify the resistance to cyberattacks.

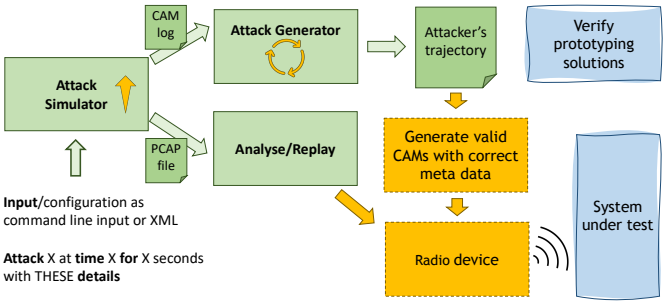


Fig. 1. Overview of the CyberSAGE framework.

The simulation of attacks in VANETs allows us to model various attacks and corner cases, however, sharing such simulations requires time for installation and configuration as well as sufficient computational resources to run them. Therefore,

CyberSAGE trains the attack generator based on the simulated data so that attack behaviours from simulation and/or realistic VANET attacks can be extracted and re-generated. The attack behaviour can be further used to verify the correct behaviour of prototypes when being exposed to attacks and to train other machine learning models for misbehaviour detection. CyberSAGE is under development with existing components marked green in Figure 1. A parser is planned to generate valid CAM messages that allows more efficient testing of VANET components to accommodate a wide variety of attacks.

The following part describes the CyberSAGE process in more detail. In Section IV, the attack simulator is presented, followed by a feasibility study of a GAN-based VANET attack generation in Section V and evaluation in Section VI.

IV. Attack Simulator

The VANET simulator in CyberSAGE consists of three components: a network simulator, a mobility simulator and the middleware which implements the vehicle behaviour and cooperative actions. The network simulator is responsible for simulating the network characteristics of each node, e.g., radio communication stacks. Mobility simulators, such as SUMO [20], simulate the roads, the individual vehicles as well as their behaviour in traffic. This includes the road topology, speed limits and driving lanes. The middleware links both simulators and allows the implementation of vehicle behaviour.

We have identified 11 VANET simulation frameworks (i.e., [5], [6], [8], [21]–[26]). Note, Eclipse MOSAIC [8] provides two versions, *Essentials* and *Extended*. Three of these eleven VANET simulation frameworks are using the ETSI ITS-G5 network stack and are still actively developed, namely Artery [22], Eclipse MOSAIC (Essential/Extended) [8] and ezCar2x [5]. CyberSAGE integrates exCar2x as the VANET simulation environment since it provides a detailed implementation of ETSI ITS communications stack including compliant CAM and DENM messages, and uses MIT, EPL or GNU GPL v2 licensed software only.

To summarize, the attack simulation framework comprises (i) ezCar2x for implementing normal and attack behaviour, (ii) ns-3 [10] for simulating the network, and (iii) SUMO [20] for traffic simulation. An overview is provided in Figure 2. The simulation is coordinated through an ns-3 script, which defines and instantiates the nodes, installs the ezCar2x applications and connects to SUMO via the Traffic Control Interface (TraCI) [20]. The roads, the environment (e.g., speed limits) and the default routes of the vehicles are defined through SUMO configurations. In the remainder of this section, we describe the overall structure of how we implemented malicious/attack behaviour in ezCar2x and further outline which attacks we have implemented.

The ezCar2x application is configured through a configuration file which allows setting the vehicles to act as an attacker or benign vehicle, the travel behaviours such as the route they should take (straight or the bypass road), and whether they should drive in a platoon, as well as attack scenarios such as which attack, when it should be performed, and

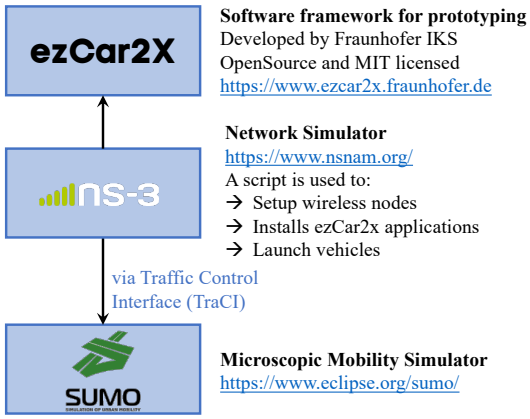


Fig. 2. Overview of the simulation environment comprising ezCar2x, ns-3, and SUMO.

attack specific parameters. This configuration is then loaded by the ns-3 script which installs the ezCar2x application. To implement certain spoofing attacks, i.e., sending incorrect information in CAM messages, it is also needed to implement additional functions for CAM generation. This can be done by modifying the existing implementation of CAM generation in *EtsiCaBasicService* in ezCar2x.

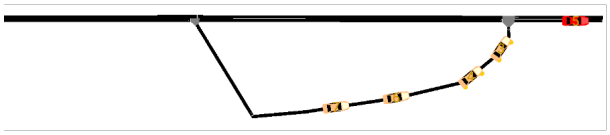


Fig. 3. An example of a simulated scenario, where all vehicles take the bypass road and the attacker spoofs a lower speed than currently driving.

Figure 3 shows the map of the SUMO simulation with a main road and a bypass road. In the given example the attacker spoofs a vehicle speed less than its current speed and thus slows down the vehicles driving behind the attacker.

In the simulation, we consider cases where the attacker has successfully compromised the vehicle or the V2X module allowing them to transmit arbitrary, yet valid CAM and DENM messages. As a first step, we have implemented four variations of spoofing attacks aiming to cause disruption in the traffic flow. The implemented attacks are the following.

Platoon Speed Spoofing. The attacker spoofs its own vehicle's speed in the CAM messages. The vehicles in the platoon are following the attacker and thus slowing down as they merely adjust their speed according to the received CAMs. In the simulation, one can choose how many speed spoofing attacks should be performed, which speed differences to the actual speed should be spoofed, and the timing.

Roadwork Spoofing. The attacker sends wrong information via DENM telling the other vehicles that there is a roadwork ahead. Therefore, vehicles change their routes and take the bypass road leaving the main road empty for the attacker.

Electric Emergency Brake. The attacker sends an incorrect DENM telling that the vehicle performs an emergency break

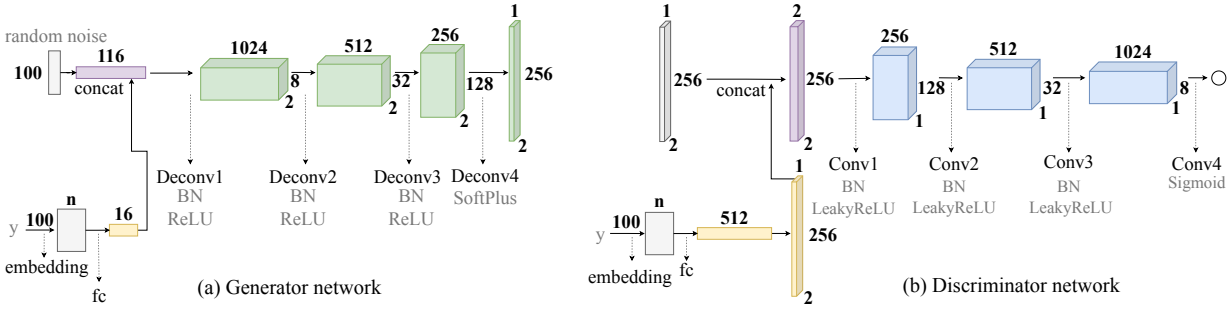


Fig. 4. The model architecture for the generator and discriminator network.

causing the vehicles behind the attacker to brake.

Sybil Attack. The attacker spoofs a number of vehicles with their own identities such that the attacker may convince the surrounding vehicles that the information (e.g., DENM warning that ice is on the road) is indeed correct.

These attacks highlight that information needs to be verified with other sensors (e.g., radar), and to perform plausibility checks as recommended in ETSI TS 102 731. Furthermore, the implementation of these simpler attacks shows that ezCar2x can be used for simulating cyberattacks.

V. Attack Generator

This section describes how we generate synthetic data using GANs. We begin the section with the preliminaries, then we present the chosen model architecture and explain how we pre-process data obtained from the attack simulation (see Section IV) as the input to the model.

A. Preliminaries

Deep generative networks are generative models that can learn the mapping from a simple latent distribution to complex data distribution and generate synthetic plausible data. GANs are such deep generative models that employ *a representation* $\rightarrow$ *generator* $\rightarrow$ *discriminator* architecture. Given a latent representation, typically sampled from a standard normal distribution $z \sim N(0,1)$, a generator G maps z to a data space while the discriminator assigns the probability $y = D(x) \in [0,1]$ of x being a sample from the training data and probability $1-y$ of x being an output of the generator $x = G(z)$. Formally, the objective function to be optimised is the following:

$$\mathcal{L}_{GAN} = \log D(x) + \log(1 - D(G(z)) \tag{1}$$

with respect to D/G. Despite wide applications of GANs and continuous development, the training of GANs poses some challenges such as mode collapse and non-convergence ([11], [27], [28]).

B. Attack design and dataset

To generate synthetic data of VANET attacks, the first step is the simulation of the speed spoofing attack and data collection. Since we are exploring the feasibility of generating plausible synthetic VANET attack data, we begin with simpler attacks that can be represented as a set of discrete or continuous variables suitable for the input to the deep learning model.

The analysed data from the simulation (see Section IV) contains fixed-length vectors of vehicle IDs, CAM speed, ground truth speed, heading, and the longitude and latitude. The data contains seven vehicles, where six vehicles are benign/normal vehicles, and one vehicle is the attacker who spoofs their speed in the CAMs. In the training data, we include 256-sample long sequences of the CAM speed and actual speed so that the input data is of size 256×2. Fixing the length and aligning the sequences allows us to treat them like images and use a convolutional layer. Examples of training data can be found in Figure 5. It shows that there is a speed spoofing attack with one speed change (the attacker's reported CAM speed is 4m/s lower between sample 80 and 130) in (a) whereas (b) and (c) show two respectively three speed changes indicated with arrows.

Following the method, we create two types of data used in the experiments:

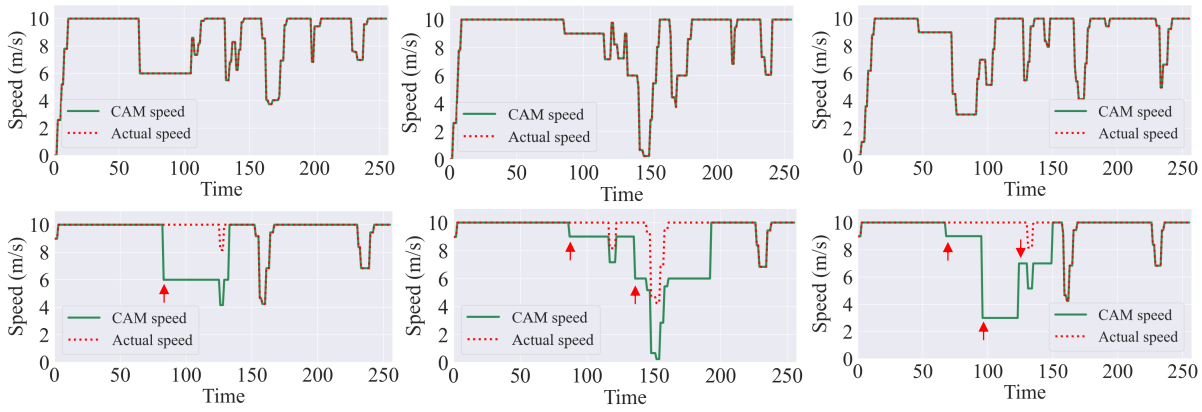
- **Data A** contains the actual speed and the CAM speed under one speed spoofing attack.
- **Data B** contains samples with one, two or three speed spoofing attacks in one sequence.

The last step is to specify the data labels. Since we aim at controlling what synthetic data we generate, either of normal vehicles or the attacker, we define two labels for data A, normal and attack, and four labels for data B, normal, and one, two, or three attacks. We train the model on each of these two datasets to see if the model is capable of capturing and then generating different patterns of speed sequences.

C. Choosing the model architecture

In the proof-of-concept, we experiment with DCGAN proposed in Radford et al. [29]. We implement the conditioned version of DCGAN and adapt it to the dimension of our input data with additional adaptations that help us make the model stable during the training. By trial and error in architecture selection, the generator and discriminator are designed as follows.

Design of the generator. The generator consists of a deconvolutional neural network composed of four layers. It expands the dimension of a random vector z of the size 1×100 to



(a) One speed spoofing attack (b) Two speed spoofing attacks (c) Three speed spoofing attacks

Fig. 5. Simulated sequences of the speed spoofing attack. **Top**: speed of normal vehicles, **Bottom**: speed of the attack vehicle.

a data sample $\hat{x}$ of the size $1 \times 256 \times 2$. The input to the generator also contains a class label which is passed through an embedding layer with a size of 100. This means that each of the class labels will be mapped to a different 100-element vector representation. The output of the embedding layer is then passed to a fully connected layer with a linear activation.

Design of the discriminator. The discriminator consists of a convolutional neural network composed of four layers. The discriminator takes the input x of the size $1 \times 256 \times 2$ and outputs 0 or 1. As in the generator, the input also contains a class label that is passed through an embedding layer to map it to a 100-element vector and then it is passed to a fully connected layer with a linear activation. Finally, it is resized to $1 \times 256 \times 2$ to match the size of the input data. The new vector of the class label and the input matrix are concatenated into a final matrix of the size $2 \times 256 \times 2$.

The model architectures of the generator and discrimination networks are presented in Figure 4. To improve the stability of the model, we apply various strategies from literature. First, Goodfellow et al. [11] stated that early in the learning the part of Eq. 1, namely $\log(1 - D(G(z)))$, saturates, which might be a reason for an unstable model. Following this, instead of training G to minimise $\log(1 - D(G(z)))$, we train G to maximize $\log D(G(z))$. Another strategy is to construct different mini-batches for real (data from the simulator) and fake (data from the generator) samples. We have also chosen different learning rates for the Adam optimiser for the discriminator and the generator. For the generator, the learning rate is $2{\cdot}10^{-4}$ and for the discriminator, the learning rate is $2 \cdot 10^{-5}$. This way, it takes more iterations for the discriminator to converge. The momentum of 0.5 is set for both the discriminator and the generator and the batch size is 64. The recommendation of using *LeakyReLU* and *tanh* activation functions did not help in achieving training convergence. Instead, the last layer of the generator contains the *softplus* activation function which results in a more stable model and better generated data.

VI. Evaluation

We have trained the model on Data A and Data B presented in Section V-B by using the model architecture described in Section V-C. In this section, we discuss how well our model is capable of generating plausible spoofed speed sequences.

A. Choosing metrics for evaluation

A wide range of evaluation metrics have been proposed to evaluate the performance of GANs [30]. For example, the evaluation of GANs usually includes the visual assessment of the quality of the generated data and some evaluation metrics such as maximum mean discrepancy or Frechet inception distance.

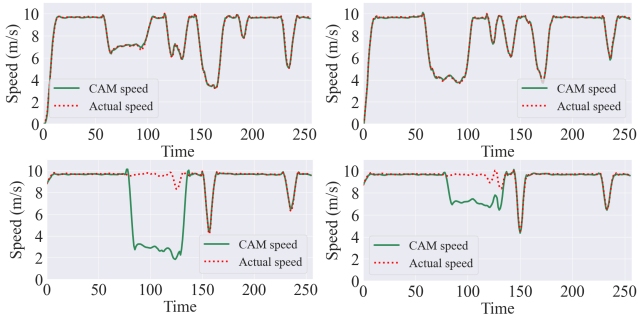


Fig. 6. Examples of generated speed sequences using the model trained on Data A. **Top**: speed of normal vehicles, **Bottom**: speed of the attack vehicle.

In this work, we first evaluate the results using the visual assessment of the generated speed sequences. The main criteria, which we use to evaluate whether the model is capable of generating plausible attacks, is that the spoofed speed and actual speed of the attack vehicle should differ at some moment, whereas the actual speed of the normal vehicles should follow the spoofed seed of the attack vehicle.

Secondly, as a qualitative assessment, we perform *k-means clustering* to show that training speed sequences and synthetic speed sequences cannot be distinguished as it is done in Zhang et al. [18]. We generate 1000 speed sequences of synthetic

data and randomly choose 1000 speed sequences from real (simulated) data. We first fit the k-mean clustering on the real speed sequences and predict the cluster labels on synthetic data. Then, we fit the k-means on the synthetic data and predict the cluster labels on the simulated data. Finally, we combine real speed sequences and generated sequences in one dataset and randomly split them into training and testing sets. The number of clusters is two as we consider two ground truth labels: *normal* and *attack* vehicles. Ideally, the performance of k-means should be the same in both cases. We evaluate the performance of clustering with an Adjusted Rand Index (ARI). We repeat each experiment 100 times, each time randomly shuffling the data samples, and then obtaining the final ARI by averaging the ARIs. An ARI value close to 0.0 corresponds to random labelling and an ARI value close to 1.0 indicates the highest corresponding to the ground truth labels.

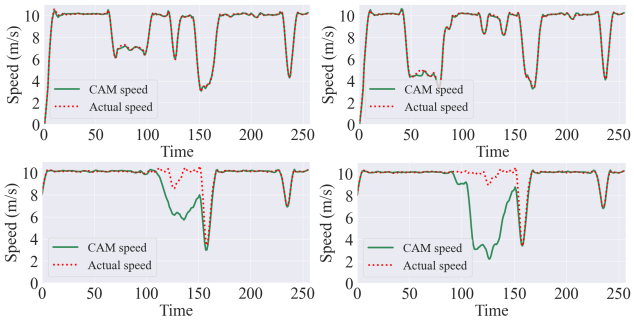


Fig. 7. Examples of generated speed sequences with one speed spoofing attack using the model trained on Data B. **Top**: speed of normal vehicles, **Bottom**: speed of the attack vehicle.

B. Results and discussion

First, we visually assess the quality of the generated speed sequences on the model trained on Data A. Note that we applied a Savitzky-Golay filter to smooth generated speed sequences. Examples of generated samples can be seen in Figure 6 for both normal vehicles (members of the platoon) in the top row and for the attack vehicle (the head of the platoon) in the bottom. As discussed earlier in this section, whether the spoofed speed sent via CAM differs from the actual speed would be considered a qualitative metric. From the figures, we can see that both, the speed sequences of the normal vehicle and the attack vehicle, meet this condition. In regard to the k-means clustering, Table I shows the results for different combinations of training and test datasets. We can see that we obtain the best clustering result for the model trained on Data A when we mix the simulated data and generated data, and when the training set is the simulated data. The ARI is slightly lower for the case when k-means is trained on the synthetic data due to the presence of noise in the synthetic data.

The next experiment is done on Data B. Figure 7 shows examples of generated sequences based on the samples with one speed spoofing attack. We can see in the speed sequences of normal vehicles (top) that the speed transmitted via CAM

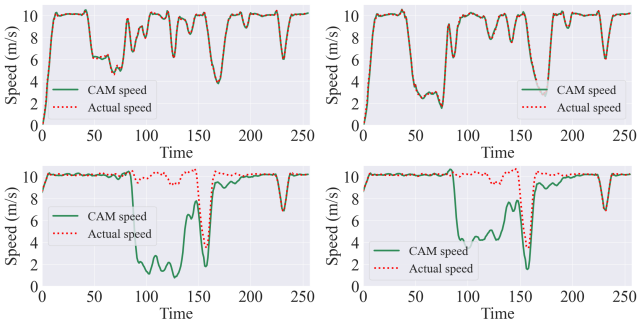


Fig. 8. Examples of generated speed sequences with two speed spoofing attacks using the model trained on Data B. **Top**: speed of normal vehicles, **Bottom**: speed of the attack vehicle.

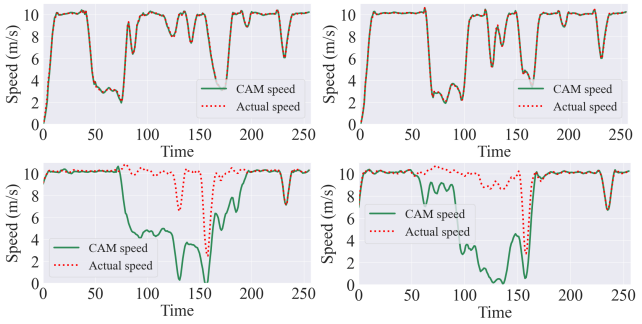


Fig. 9. Examples of generated speed sequences with three speed spoofing attacks using the model trained on Data B. **Top**: speed of normal vehicles, **Bottom**: speed of the attack vehicle.

matches the actual speed, while the speed sequences of the attack vehicle (bottom) show variations between spoofed speed via CAM and actual speeds. This indicates that the model can capture the difference between benign/normal and attack vehicles. Figure 8 and Figure 9 illustrate examples of generated speed sequences for the samples with two speed spoofing attacks and three speed spoofing attacks, respectively. The results show a clear difference between the spoofed speed (green) and the actual speed (red) during an attack which indicates that the model captures the two main types of vehicles. Another observation is that the number of speed changes is not tightly correlated with the number of spoofing attacks. Based on how we simulate the training data, we should see the different number of speed changes in the spoofed speed sequence. However, more than one speed spoofing attack is not entirely obvious from the figures since vehicles need to slow down when taking the right turn or when driving in the curve as it is illustrated in Figure 3. The result of k-means clustering for the data generated with the model trained on Data B can be found in Table I.

VII. Conclusion

Cybersecurity is vital for the successful introduction of connected and autonomous vehicles. Different attacks must be detected at an early stage to prevent potential consequences. Machine learning (ML) has been widely used for the design of

Train	Test	ARI (Data A)	ARI (Data B)
Simulated	Synthetic	1.0	1.0
Synthetic	Simulated	0.98	0.84
Mixed	Mixed	1.0	0.99

TABLE I
k-MEANS CLUSTERING PREDICTION RESULTS. THE SYNTHETIC DATA IS GENERATED WITH THE MODEL TRAINED ON DATA A RESPECTIVELY DATA B.

various intrusion or misbehaviour detection systems. Effective ML-based solutions for intrusion detection systems and their testing require a large amount of training data related to different attack scenarios. The lack of such real-world VANET attack traffic is limiting the development of efficient methods for improving cybersecurity, especially ML-based methods. In this paper, we present CyberSAGE, a framework that facilitates cybersecurity research. The framework leverages multiple simulators for data generation and collection, followed by synthetic data generation based on GANs. We have shown based on the speed spoofing attack scenario that we can generate plausible synthetic data.

CyberSAGE is in the starting phase with a proof of concept with several future research directions. An obvious direction is the inclusion of multiple and more complex attacks to accommodate the increasing complexity when more vehicles are connected. Another direction is the improvement of deep generators so that we can generate the VANET attacks that follow a certain structure needed for testing. New architectures following the latest development in artificial intelligence will be considered and integrated.

REFERENCES

[1] C. Englund, L. Chen, A. Vinel, and S. Y. Lin, *Future Applications of VANETs*. Cham: Springer International Publishing, 2015, pp. 525–544. [Online]. Available: https://doi.org/10.1007/978-3-319-15497-8_18

[2] H. Hasrouny, A. E. Samhat, C. Bassil, and A. Laouiti, "VANet security challenges and solutions: A survey," *Vehicular Communications*, vol. 7, pp. 7–20, 2017. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2214209616301231

[3] F. Gonçalves, B. Ribeiro, O. Gama, A. Santos, A. Costa, B. Dias, J. Macedo, and M. J. Nicolau, "A systematic review on intelligent intrusion detection systems for VANETs," in *2019 11th International Congress on Ultra Modern Telecommunications and Control Systems and Workshops (ICUMT)*, 2019, pp. 1–10.

[4] A. Vahidi, T. Rosenstatter, and N. I. Mowla, "Systematic evaluation of automotive intrusion detection datasets," in *Proceedings of the 6th ACM Computer Science in Cars Symposium*, ser. CSCS '22. New York, NY, USA: Association for Computing Machinery, 2022. [Online]. Available: https://doi.org/10.1145/3568160.3570226

[5] Fraunhofer IKS, "ezCar2x: Streamlined Development of Networked Vehicle Applications," https://www.ezcar2x.fraunhofer.de/en.html, cited April 2022.

[6] J. Kamel, M. R. Ansari, J. Petit, A. Kaiser, I. Ben Jemaa, and P. Urien, "Simulation framework for misbehavior detection in vehicular networks," *IEEE Transactions on Vehicular Technology*, 2020.

[7] S. Iqbal, P. Ball, M. H. Kamarudin, and A. Bradley, "Simulating malicious attacks on VANETs for connected and autonomous vehicle cybersecurity: A machine learning dataset," 2022. [Online]. Available: https://arxiv.org/abs/2202.07704

[8] Eclipse Foundation, "Eclipse Mosaic - a multi-domain and multi-scale simulation framework for connected and automated mobility," https://www.eclipse.org/mosaic/, cited April 2022.

[9] 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*, ser. SIN '18. New York, NY, USA: Association for Computing Machinery, 2018.

[10] nsnam.org, "ns-3 network simulator," https://www.nsnam.org/, cited May 2022.

[11] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair *et al.*, "Generative adversarial nets," in *Advances in Neural Information Processing Systems*, Z. Ghahramani, M. Welling, C. Cortes, N. Lawrence, and K. Weinberger, Eds., vol. 27. Curran Associates, Inc., 2014.

[12] E. Seo, H. M. Song, and H. K. Kim, "GIDS: GAN based intrusion detection system for in-vehicle network," in *2018 16th Annual Conference on Privacy, Security and Trust (PST)*. IEEE, aug 2018. [Online]. Available: https://doi.org/10.1109%2Fpst.2018.8514157

[13] J. Shu, L. Zhou, W. Zhang, X. Du, and M. Guizani, "Collaborative intrusion detection for VANETs: A deep learning-based distributed SDN approach," *IEEE Transactions on Intelligent Transportation Systems*, vol. 22, no. 7, pp. 4519–4530, 2020.

[14] D. Parthasarathy, K. Bäckström, J. Henriksson, and S. Einarsdóttir, "Controlled time series generation for automotive software-in-the-loop testing using GANs," 2020. [Online]. Available: https://arxiv.org/abs/2002.06611

[15] D. Parthasarathy and A. Johansson, "SilGAN: Generating driving maneuvers for scenario-based software-in-the-loop testing," 2021. [Online]. Available: https://arxiv.org/abs/2107.07364

[16] N. M. E. Listo Zec, H. Arnelid, "Recurrent conditional GANs for time series sensor modelling." the Time Series Workshop at International Conference on Machine Learning, january 2019.

[17] E. Brophy, Z. Wang, and T. E. Ward, "Quick and easy time series generation with established image-based GANs," 2019. [Online]. Available: https://arxiv.org/abs/1902.05624

[18] C. Zhang, S. R. Kuppannagari, R. Kannan, and V. K. Prasanna, "Generative adversarial network for synthetic time series data generation in smart grids," *2018 IEEE International Conference on Communications, Control, and Computing Technologies for Smart Grids (SmartGridComm)*, pp. 1–6, 2018.

[19] J. Yoon, D. Jarrett, and M. Van der Schaar, "Time-series generative adversarial networks," *Advances in neural information processing systems*, vol. 32, 2019.

[20] P. A. Lopez, M. Behrisch, L. Bieker-Walz, J. Erdmann, Y.-P. Flötteröd, R. Hilbrich, L. Lücken, J. Rummel, P. Wagner, and E. Wießner, "Microscopic traffic simulation using SUMO," in *The 21st IEEE International Conference on Intelligent Transportation Systems*. IEEE, 2018. [Online]. Available: https://elib.dlr.de/124092/

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

[22] R. Riebl, "Artery – V2X simulation framework," http://artery.v2x-research.eu, cited April 2022.

[23] Rubinet Lab, ECE Department, University of California, Davis, "VENTOS: VEhicular NeTwork Open Simulator," https://maniam.github.io/VENTOS/, cited April 2022.

[24] EstiNet Technologies Inc., "EstiNet VANET Add-on," https://www.estinet.com/ns/?page_id=21146, cited April 2022.

[25] TETCOS, "NetSim: Model - Predict - Validate," https://www.tetcos.com/, cited April 2022.

[26] M. Malinverno, F. Raviglione, C. Casetti, C.-F. Chiasserini, J. Mangues-Bafalluy, and M. Requena-Esteso, "A multi-stack simulation framework for vehicular applications testing," in *Proceedings of the 10th ACM Symposium on Design and Analysis of Intelligent Vehicular Networks and Applications*, ser. DIVANet '20. New York, NY, USA: Association for Computing Machinery, 2020, p. 17–24.

[27] H. Chen, "Challenges and corresponding solutions of generative adversarial networks (GANs): a survey study," in *Journal of Physics: Conference Series*, vol. 1827, no. 1. IOP Publishing, 2021, p. 012066.

[28] Z. Wang, Q. She, and T. E. Ward, "Generative adversarial networks in computer vision: A survey and taxonomy," *ACM Computing Surveys (CSUR)*, vol. 54, no. 2, pp. 1–38, 2021.

[29] A. Radford, L. Metz, and S. Chintala, "Unsupervised representation learning with deep convolutional generative adversarial networks," *arXiv preprint arXiv:1511.06434*, 2015.

[30] A. Borji, "Pros and cons of GAN evaluation measures: New developments," *Computer Vision and Image Understanding*, vol. 215, p. 103329, 2022.