On Vehicular Data Aggregation in Federated Learning

A Case Study of Privacy with Parking Occupancy in Eclipse SUMO

Levente Alekszejenkó[1],
and Tadeusz Dobrowiecki[1]

[1]Department of Measurement and Information Systems, Budapest University of Technology and Economics, Budapest, Hungary
*Correspondence: Levente Alekszejenkó, alelevente@mit.bme.hu

Abstract: Vehicular federated learning systems will be beneficial to predicting traffic events in future intelligent cities. However, they might leak private information upon model updates. Hence, an honest but curious server could infer private information, such as the route of a vehicle. In this study, we elaborate on the nature of such privacy leakage caused by gradient sharing. With a simulated scenario, we focus on determining who is in danger of privacy threats and how successful a route inference attack can be.

Results indicate that vanilla federated learning exposes intra-city and commuter traffic to successful location inference attacks. We also found that an adversarial aggregator server successfully infers the moving time of vehicles traveling during low-traffic periods.

Keywords: Vehicular Federated Learning, Location Privacy, Deep Leakage From Gradients

1 Introduction

As the newest vehicles have more and more sensors, such as cameras, ultrasonic, radar, and lidar sensors, they can measure various phenomena along their route. Most of these measurements supply valuable information for other vehicles or to the infrastructure maintainer. Hence, sharing these data will be important in a short time.

Since the manufacturers might use various sensors, the vehicles can have different sensing and processing capabilities. Moreover, the vehicular communication also limits the amount of exchangeable information. This information shall be up-to-date to provide adequate input for, e.g., navigation algorithms. However, providing direct measurement data reveals the vehicles' trajectories, thus making a serious privacy breach.

There is a machine learning method called federated learning (FL) [1] that might solve the above challenges. FL is a decentralized learning method that is free of raw data sharing. A traditional FL system consists of a central aggregator *server* and many *participants* (e.g., vehicles). The server maintains a global model that the participants can download. The participants can use the obtained model, e.g., to infer the current

state of the road network [2]. The participants can also train and update this model with their local data. The server periodically collects and aggregates these updates. The traditional aggregation method is the *FedAvg* algorithm that averages the participants' local models weighed by the size of their local dataset used for training. At the end of the communication rounds, the server sends the updated aggregated global model back to the participants. Hence, the participant will have an up-to-date version of the model. Moreover, due to the regular updates, the model continuously adapts to the actual state of the road network. As the server and the participants send the model, i.e., a matrix of weights of a neural network, this is a compressed version of the knowledge obtained by the participants; therefore, it can reduce the usage of the communication channel compared to sharing raw data.

In an ideal case, this knowledge exchange method provides privacy for the participants as they do not need to share their measurement data. However, in practice, this is not the case. By comparing the global model to a local update, the aggregator server might gain information about the data of a participant vehicle. We call this phenomenon the *gradient leakage*.

An honest but curious[1] server might exploit the gradient leakage to perform a localization attack to infer one's route and moving time. Since this information is privacy sensitive, the FL technique poses a cyber security risk. This paper focuses on analyzing the risks caused by FL. To this end, we have created a small virtual town and measured the parking lot occupancy. This research aimed to discuss the following questions:

1. Who is in danger due to the leaking gradients during the training of a FL system?
2. How successfully can an honest but curious server track a vehicle?

Besides answering the above questions, this study contributes to parking lot simulation in Eclipse SUMO [3] by implementing a parking activity creator tool and presenting its usage via a case study.

The rest of this paper includes a short literature overview in Section 2. In Section 3, we describe the scenario generation and process of the simulation. Based on the obtained simulation results, we assessed the leaking gradient problem. Section 4 summarizes our findings, and finally, Section 5 concludes this paper.

2 Related literature

Finding a vacant parking lot near our destination is challenging in dense urban environments. If drivers had information about the current parking lot occupancy rates, they would be more successful in parking place finding. It would also significantly reduce the searching time [4]; therefore, intelligent transportation networks would benefit from a parking occupancy prediction system. In our research, we simulated parking activities by Eclipse SUMO as it can provide large-scale parking lot simulations. For example, the Python Parking Monitoring Library *(PyPML)* [5] provides routines to measure and control parking lot usage in a SUMO simulation.

Modern vehicles equipped with, e.g., cameras can collect information about parking lot occupancies, but to utilize the gathered data, they should share it over vehicular communication techniques (Vehicle-to-Everything, V2X). Besides regular engineering

---

[1] An honest but curious server rigorously follows the FL protocol without maliciously manipulating the model. However, it tries to obtain as much information about the participant vehicles as possible.

requirements, vehicular communication systems shall also respect the drivers' privacy [6], especially their location data.

In recent years, FL [1] became popular because it provides a communication efficient way to exchange compound information. Moreover, in ideal cases, it can also preserve the clients' privacy. Consequently, FL has high potential in vehicular use cases [7]. FL can also help to comply with the General Data Protection Regulation (GDPR) of the European Union, but there is a trade-off between privacy protection, and model utility in FL systems [8].

One of the main concerns in FL is the *gradient leakage* [9], which means that an adversarial FL aggregation server might infer properties of the training set of an FL client by observing its updates in the weight matrix, or the gradients that are proportional to them. Our previous study demonstrated that vehicular FL also leaks gradients, which reveals the drivers' private information such as location and moving time [2]. In this paper, we intend to elaborate more on these risks. Following the taxonomy of [10], we assume that a vehicular, horizontal FL system contains a passive, honest-but-curious aggregator server. Consequently, it has (plaintext) access to the clients' model updates. This server performs a single-shot, targeted, passive privacy attack against a client vehicle during the training time. The attack aims to obtain the targeted vehicle's location and moving time information.

3 Simulation

To address the problem of leaking gradients, first, we have to collect training data. Therefore, we simulated the vehicle movements in a small town to obtain realistic parking lot occupancy data. For this simulation, we shall define an appropriate scenario. We assumed generated scenarios encourage reproducibility; therefore, we tried to use SUMO's generator tools whenever possible. The following sections describe their parametrization, design decisions, and a new tool that creates parking activities based on predefined *trips*.

3.1 Road network

Firstly, we created a random road network with the `netgenerate` tool. We parametrized the tool to insert edges 30 times into the net and set up an actuated traffic light.

After that, we drew a land use plan. The generated road network contained a community of edges on the south part of the map, which we intended to treat as a village-like suburb of the town to the north consisting of a couple of residential buildings and a few commercial amenities forming small mixed-usage zones (see Figure 1). Moreover, mixed zones with various housing and workplace densities form the core area of the town, together with some pure residential zones and a small commercial zone.

Additionally, we shall also define parking lots in the road network. As in a typical small town, we permitted on-street parking on each road. Unfortunately, on-street parking cannot serve the whole parking demand. Hence, we added a handful of off-street parking facilities into the scenario.

We can derive the rest of the population's activities in the simulated town from the land use plan. However, this plan does not describe the place of educational facilities, which generate significant traffic. Consequently, we placed 1 kindergarden, 3 elementary, and 1 middle school on the map.
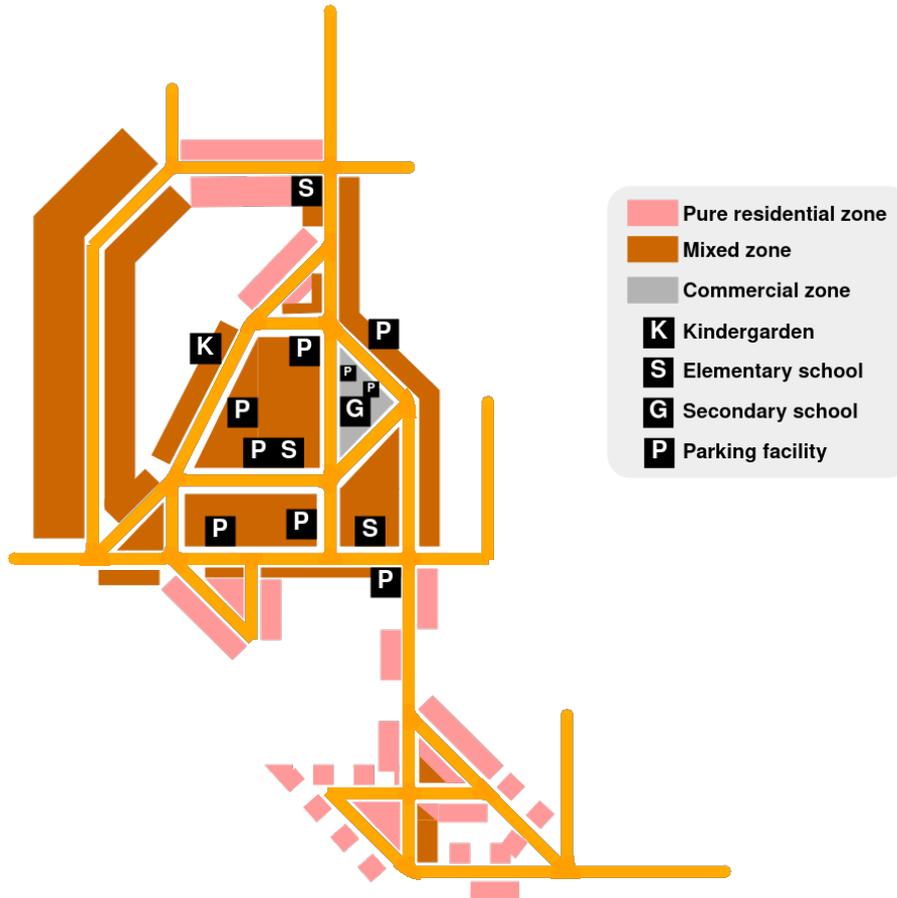
Figure 1. *Road network and land use of the generated town.* [2]

Following the German coloring scheme of land use plans [11], Figure 1 illustrates the static elements of the simulation scenario.

3.2 Transportation activities

Besides the static elements of the road network, a traffic scenario requires the definition of the traffic demand. To this end, Eclipse SUMO has a tool called `activitygen`. The input of this tool includes a statistics file describing the town's population.

In our scenario, the simulated town has 10,000 inhabitants within 3,500 households having a 0.58 $\frac{vehicles}{household}$ car posession rate. As these data and the road network correspond to a small town, we defined the age category following the demographic distribution of the average European Union intermediate (rural) region [12] as summarized in Table 1. We assumed that 10%-10% of the working age group (e.g., shopkeepers and employees of some services) begin working respectively at 6:00 and 7:00. These people usually leave their workplace around 14:00. Assumably, the majority (50%) of the population works (including children going to school) between 8:00 and 16:00. However, there might be a minority (30%) who start working at 9:00. We suspect that many of them work in flexible time; resulting in a varied closing time around 17:00 and 18:00, see Figure 2. Besides working, the population sometimes (with a 0.1 rate) also participates in free time activities.

---

[2]In mixed zones, the residential and commercial area densities vary. The placement of the amenities along a road only serves illustration goals; for exact details, see sources on Github.

Table 1. *Age groups and their population in the simulated town.*

Age group	#people in the town
0–15 years	1490
15–19 years	520
19–64 years	5830
64–79 years	1540
79–100 years	620
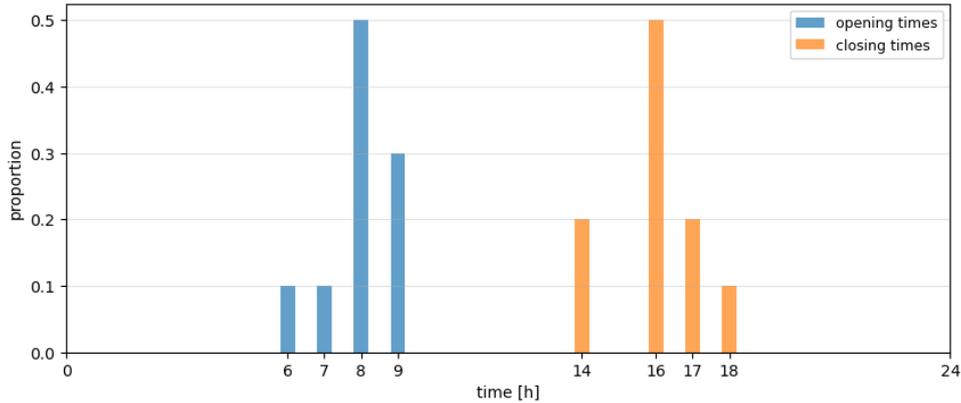
Figure 2. *Working opening and closing hours of the population of the simulated town.*

200 people leave the town daily, and 1000 persons form an incoming commuter traffic. Moreover, we added some random, uniform background traffic through the city (with 0.1 rate). Consequently, there are three vehicle types in the simulated scenario:

- *Household* vehicles reside in the town, generating a parking demand even at night. They usually depart in the morning, do some activities during the day, and return home in the evening.
- *Commuter* vehicles come to the town every morning. Then they do some activities and leave the town usually in the evening.
- *Random* vehicles go through the town without any recurrence or taking any activities.

3.3 Parking simulation

By default, in Eclipse SUMO, the road network is empty at the beginning of a simulation, and all parking lots are vacant. Consequently, the first simulated vehicles can easily find parking spaces at their destination, but as the roads get more and more packed, it will harden the challenge of parking lot seeking. To mitigate this transient, we ran burn-in simulations for 4 simulated days. After this burn-in phase, we can make measurements assumed to be free of such transient states.

However, SUMO's `activitygen` tool cannot handle activities that simulate the parking lots or recurrent activities. Hence, we implemented a new tool called `parking_activities`[3]. This tool, written in Python, processes the `xml` output file of `activitygen` and enhances the trips of the different vehicle types in the following manner:

- *Household* trips: Household vehicles will repeat the activities for days that the `activitygen` tool created for them. Our tool collects all individual movements of a *household* vehicle and merges them into one *trip* that originates and terminates

---

[3]Available on GitHub.

at its home location. Within this trip, `parking_activities` fills the time between the original movements with *stops* on the on-street parking lot at the place of the original activity. Our `parking_activities` tool also repeats these activity chains for an input number of days. Between two days, the *household* vehicles park at their home location for random time drawn from a normal distribution. The parameters of this distribution ensure that approximately 95% of the *household* population starts their daily activity each day within a ± 15 minutes window.

- *Commuter* trips: As the commuter vehicles enter and leave the simulation each day, they are easier to handle by `parking_activities`. It only adds *stops* in a parking lot at the destination of the inbound movement of the commuter vehicles. For each prescribed day, our tool repeats these activity chains of commuter vehicles with new *trip identifiers*.
- *Random* trips: The `parking_activities` tool only repeats the original random activities created by `activitygen` for each presribed day.

In this way, Eclipse SUMO shall handle the moving time uncertainties for *commuter* and *random* vehicles, and `parking_activities` is responsible for simulating the random departure time of *household* vehicles.

Finally, the `parking_activities` script saves the modified trips into a new file that will follow the definition of `activitygen`'s output. Consequently, we can feed this enhanced file into SUMO's `duarouter` tool to generate traffic for the simulation scenarios.

The simulated vehicles behave according to the default parameters in SUMO, and, during their movements, they measure (via subscriptions) the $\mathcal{O}(p, t)$ occupancy rate of parking lot p at the t measurement time. We used a 50 m measurement range that is a feasible visual sensor range or a distance within the performance of neither of the vehicular communication techniques degrades significantly [13]. We used a 1 s simulation step size and 1 Hz measurement frequency. In our experiments, SUMO controlled the parking lot simulation according to the predefined activity chains. If the predefined parking lot was already occupied, parking area rerouters helped the vehicles find vacant parking spots nearby.

5 simulated day-long simulations (with their corresponding 4 day burn-in phase) were repeated 5 times with different initial random seeds. In the following, from a machine learning point of view, we treat these results as if we had an independent random mixture of 25 days of data; despite the fact, that it came from 5 independent measurement periods containing data of 5 consecutive days.

4 Leaking gradients

Once we had obtained the simulation results, we could evaluate the problem of leaking gradients as if the vehicles participated in an FL system.

4.1 Evaluation methods

To elaborate on the gradient leakage problem, we shall set up a FL system with a central aggregation server. Unfortunately, training an entire FL system is a time-consuming process. To save computational resources, we simulated the first part of the FL training process in the following way: We have selected 4300 vehicles (out of the 4644 unique simulated vehicles) to pre-train a feed-forward neural network model. We trained this model with an early stopping mechanism to ensure satisfactory prediction performance without overfitting the training data.

This pretrained neural network symbolizes a *global FL model* trained nearly to convergence. We selected 300 vehicles (from the remaining 344) and updated the pretrained network with their local data. The resulting new models correspond to the updated *models of the clients* in an FL system. Therefore, we can analyze these models for gradient leakages.

4.1.1 Location inference

To formulate the inference methods, we denote the set of measurement times (seconds of a day) by $T = [1, 2, 3, \ldots, 24 \cdot 60 \cdot 60]$ and the set of the defined parking lots in the road network by P.

To evaluate the privacy leakage in this simulated FL system, we ran inferences with the global and the local models, resulting respectively in occupancy estimates $\widehat{\mathcal{O}}_g(p, t)$ and $\widehat{\mathcal{O}}_l(p, t)$ for each $(p, t) \in P \times T$. Let us define the average location difference $\overline{\Delta}_p$ as:

$$\overline{\Delta_p} = \frac{\sum\limits_{t \in T} \left(\widehat{O}_g(p, t) - \widehat{O}_l(p, t)\right)^2}{|T|}. \tag{1}$$

After the inference phase, we evaluated the $\bar{\Delta}_p$ average location difference for each $p \in P$ parking lots and ranked them into ascending order. We assume that those parking lots differ more that were inside a vehicle's training set. To infer the vehicles' positions, we appointed the top 10 most differing parking lots as the inferred position. The rationale for this number is that if a driver moves along the least complicated pattern, a triangle in the road network, it will have data from at least 6 parking lots (from the two sides of the three streets). On the other hand, if we choose too many parking lots, we risk that the target vehicle's dataset does not even contain data from so many parkings.

Finally, to measure the success rate of the gradient leakage attack, we check how many of the 10 inferred parking lots are in the training set of the target vehicle. We refer to this count as the *positional success rate* of the honest-but-curious server.

4.1.2 Moving time inference

We assume that tracking attacks consist of both location and moving time inference. To address the latter, let us introduce a w time window. As the moving time is rather an interval than an exact moment, the moving time inference method shall try to infer the time window in which the target had moved. In this study, we used $w = 900$ s $= 15[\text{ min}]$ long time windows.

The different vehicle types have different moving patterns: the *random* vehicles move probably in 1 time window, the *commuters* at least in 2. The *household* vehicles often move in 2 time windows; however, they can be active in more or even less number of intervals. Consequently, each vehicle travels in at least one time window. Hence, we aim to infer 1 time window in which the target had moved.

To infer a moving time window, we shall resample our data set in time to correspond to the time windows. We denote the set of resampled measurement times $\tau_0 = 0, \tau_1 = w, \tau_2 = 2w, \ldots, \tau_n = \frac{24 \cdot 60 \cdot 60}{w}$ measured in seconds. Respectively, the $\widehat{O}_{t,g}$ and $\widehat{O}_{t,l}$ resampled global and local predictions will be the mean of the predictions

with the original sampling frequency aggregated over the parking lots:

$$\widehat{O}_{t,g} = \frac{\sum_{p \in P} \sum_{i=\tau_t}^{\tau_{t+1}-1} \widehat{O}_g(p,i)}{w},$$ (2)

$$\widehat{O}_{t,l} = \frac{\sum_{p \in P} \sum_{i=\tau_t}^{\tau_{t+1}-1} \widehat{O}_l(p,i)}{w}.$$ (3)

We shall aggregate the predictions over the parking lots to obtain solely time-related information. Moreover, as we mentioned, moving time is an interval, not a single moment. Therefore, we expect that the training data will create plateaus in the $\left(\widehat{O}_{t,g} - \widehat{O}_{t,l}\right)^2$ difference. To find (the beginning or the end of) one of these plateaus, we take the absolute value of the first derivative of the difference of (2) and (3). Then, we identify the $\widehat{\tau}$ moving time as the location of the maximum:

$$\widehat{\tau} = \arg\max_t \left| \frac{d}{dt}(\widehat{\mathcal{O}}_{t,g} - \widehat{\mathcal{O}}_{t,l}) \right|.$$ (4)

Figure 3 illustrates the idea behind the proposed moving time inference method.
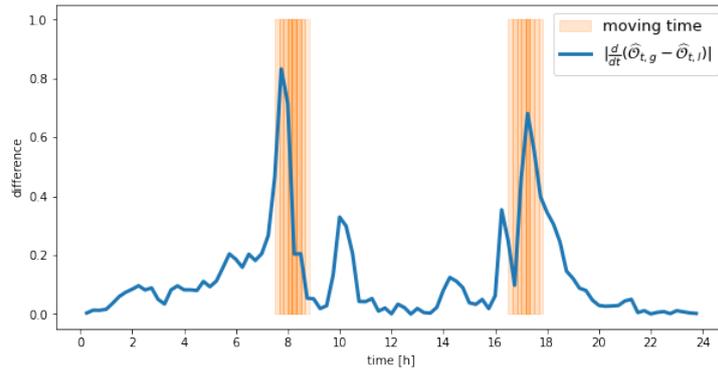
Figure 3. *Differences between the local and global model predictions in time compared to the movements of a vehicle. Moving times are colored proportionally to the frequency of the corresponding time windows in vehicle's dataset.*

To measure the success of the moving time prediction algorithm, we shall quantify its error. To this end, we look for the τ_c time window in the target's dataset that is closest to the $\widehat{\tau}$ prediction. The t_o *absolute time offset* between these two time windows is the measure of the success of the algorithm, and it is defined as:

$$t_o = |\widehat{\tau} - \tau_c|.$$ (5)

We shall note that the t_o is measured in w long time windows.

4.2 Results

The evaluation shows that the location inference method is generally successful, achieving an average of 55.1–74.1% *positional success rate* with an approximately constant standard deviation, see Table 2. Consequently, the FL system leaks significant positional information in the gradients. The results also show that *household* traffic is in

more danger than *commuter* and *random* traffic in terms of location privacy; see Figure 4. The primal cause of this difference is that *household* vehicles visit more parking lots than the other vehicle types; therefore, it is more likely to choose 10 parking lots that they have measured.

Table 2. *Means and standard deviations of the adversarial success results.*

	Positional success rate	Abs. time offset
commute	5.98 ($\pm$ 2.02)	1.83 ($\pm$ 2.21)
household	7.41 ($\pm$ 2.07)	6.26 ($\pm$ 9.39)
random	5.51 ($\pm$ 2.15)	16.78 ($\pm$ 10.74)
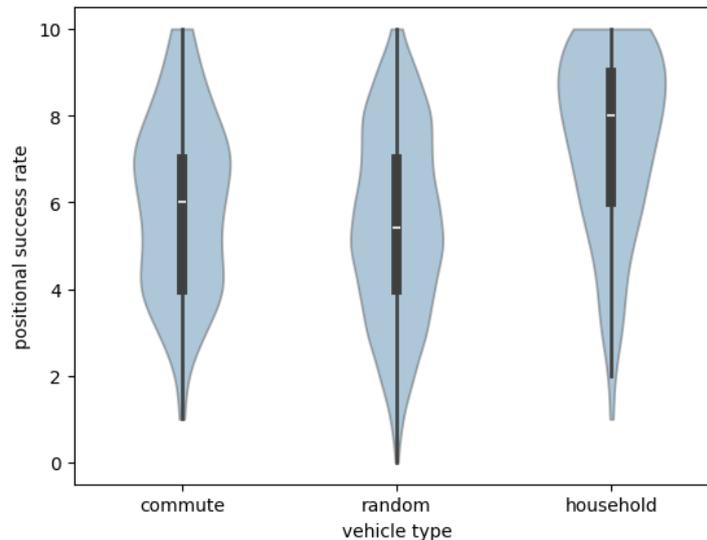
Figure 4. *Success rate of the location inference by gradient leakage.*

4.2.1 Commuter and random traffic

The location inference method results in a constant performance regardless of the vehicle types. On the other hand, vehicle type categories fundamentally influence the success of time inference according to Table 2 and Figure 5.

For the *random* traffic, the moving time inference algorithm achieves an average of 16.78 absolute time offset value corresponding to approximately 4 hours of error with an immense standard deviation. It implies that *random* vehicles do not leak measurement time information while contributing to the FL system.

On the other hand, the moving time inference method identifies the movements of the *commuting* vehicles outstandingly well, having an average absolute time offset of 1.83, which corresponds to an error of approximately 28 minutes. We might have expected these results as the *commuter* vehicles come to work in the town and leave after their shift ends, and these movements happen in a well-defined time range. Consequently, commuter vehicles leak significant information about the moving times during updating the FL model.

4.2.2 Household traffic

The *household* vehicle type creates an exceptionally long-tailed distribution in absolute time offset; see Figure 5. We suspect that we see here a mixture of two different
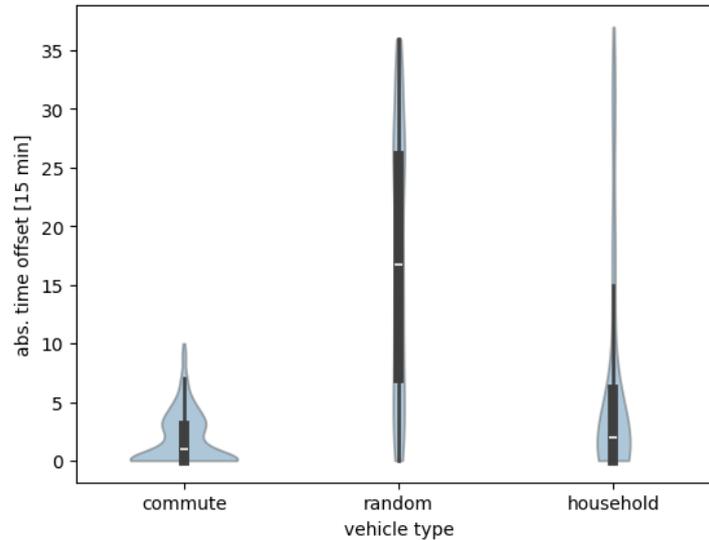
Figure 5. *Success rate of the moving time prediction by gradient leakage.*

distributions. To this end, we ran a KMeans clustering [14] to create two classes in the absolute time offset values.

Surprisingly, we found that the two clusters also differ in positional success rate; see Figure 6. Moreover, according to Figure 7, the vehicles belonging to *cluster0* leak significantly more moving time information than those belonging to *cluster1* as the inference algorithm achieves a much lower absolute time offset.
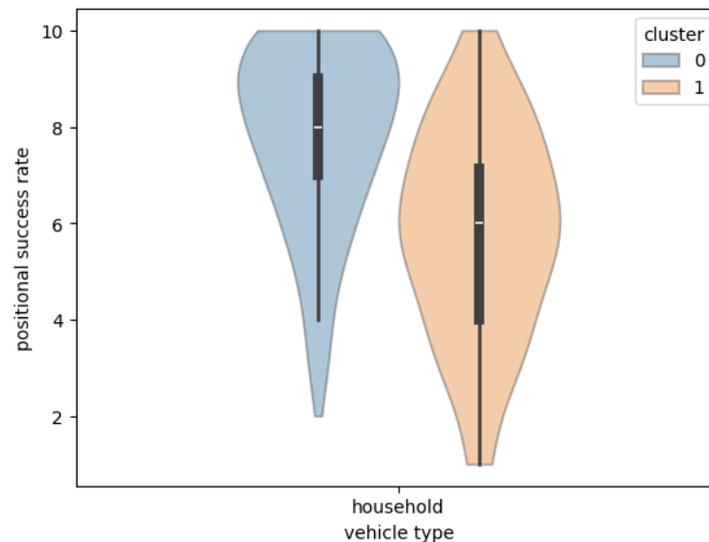
Figure 6. *Success rate of the location inference by gradient leakage in the two clusters of household traffic.*

Now, we shall investigate what separates the two clusters. We have checked the route of the vehicles of both clusters, but we did not find any significant difference. After that, we compared their moving times. Those in *cluster0*, see Figure 8a, usually travel in the morning and the evening like people with a traditional working time. On the other hand, those in *cluster1* drive at night according to Figure 8b. At night, the parking lots' occupancy is more or less fixed as only a few cars are on the roads that late. Therefore, those in *cluster1* could measure the same occupancy rate at around 20:00 and 4:00 the next day, which can confuse the moving time inference algorithm.
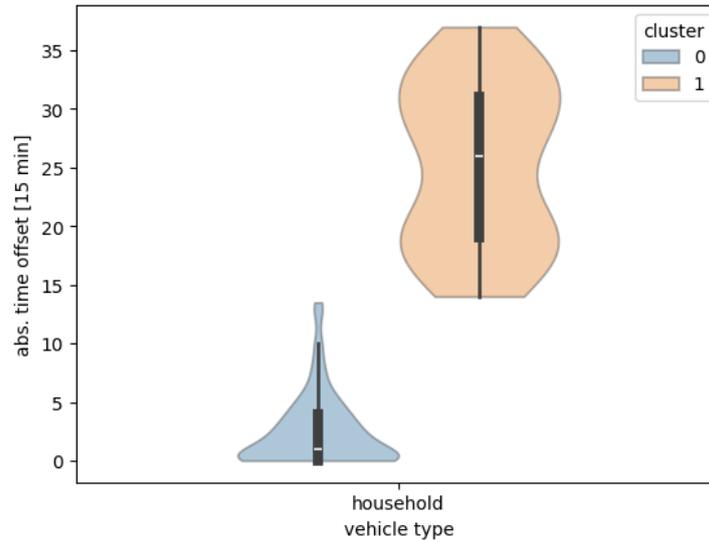
Figure 7. *Success rate of the moving time prediction by gradient leakage in the two clusters of household traffic.*
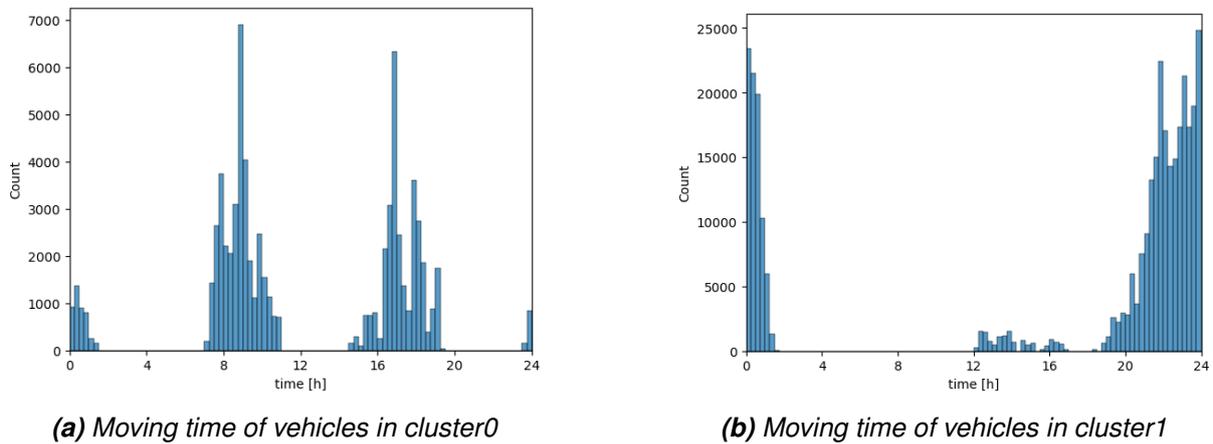
(a) Moving time of vehicles in cluster0

(b) Moving time of vehicles in cluster1

Figure 8. *Moving time of household vehicles.*

Consequently, those household vehicles that move in the morning and the evening leak more information about their moving times than those that drive in small traffic, i.e., at night.

4.2.3 Discussion

Training neural networks is a non-deterministic process. Therefore, we repeated the experiments five times to handle it. We selected another set of vehicles as trainers and targets for each experiment and ran the entire training, inference, and evaluation process. The presented values in this study are the aggregated results of these experiments.

The presented inference algorithms are heuristic and do not utilize any additional information. One may construct more sophisticated attacks against specific targets, e.g., by testing whether a parking lot is inside the target's dataset. Moreover, the moving time inference algorithm does not rely on knowledge of visited parking lots. In our implementation, the moving time inference algorithm looks only for the highest peak; however, it would be possible to check whether a specific target moved during a pe-

riod. In summary, it would be possible to create more powerful inference algorithms that would cause severe privacy leakage.

When evaluating the position inference algorithm, we shall check whether the results depend on the road network. For example, a bottleneck edge would add two on-street parkings likely visited by the vehicles. That would positively offset the positional success rate. In our road network, there is a bottleneck between the core part of the town and the small village-like area at the south of the map. As it is not a central part of the road network, only a smaller portion of the vehicles take this road. Hence, we assume its influence is marginal to the final results.

Our final observation is in connection with our previous study [2]. In the Monaco SUMO Traffic Scenario [15], each vehicle belongs to the *random* type as no one repeats its activity chain. Consequently, tracking them in time is more challenging.

5 Conclusion

We can conclude that *household* vehicles moving during daytime leak the most location information. *Commuters* leak time information in an FL system. The *random* traffic and the vehicles moving at night leak less private information in an FL training process aiming to predict parking lot occupancies.

This study also presents that *household* vehicles' of those having conventional working hours and *commuter* vehicles are easy to track in location (with approximately 80% and 60% average success rate respectively) and with small uncertainty in time (with an average of approximately 30 minutes).

Our present paper calls attention to two important observations:

1. Vanilla federated learning in vehicular communication systems leaks significant privacy-sensitive data.
2. Consequently, when designing vehicular federated learning systems, we shall provide adequate countermeasures to ensure the users' privacy also at the *application level* in the OSI model [16].

Data availability statement

To reproduce the results presented in this paper, one may use the https://github.com/alelevente/inverse-parking repository. The uploaded source codes also generate the necessary input files.

Author contributions

Levente Alekszejenkó is a Ph.D. Student under the supervision of Tadeusz Dobrowiecki. As a part of Mr. Alekszejenkó's Ph.D. research, he conducted the investigation and created the visualization presented in this paper. Both authors are responsible for the presented concepts and methodologies. Prof. Dobrowiecki also had many constructive commentaries on the manuscript written by Mr. Alekszejenkó in the pre-publication stage.

Competing interests

The authors declare that they have no competing interests.

Funding

References

[1] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y. Arcas, "Communication-efficient learning of deep networks from decentralized data," in *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics*, ser. Proceedings of Machine Learning Research, A. Singh and J. Zhu, Eds., vol. 54, PMLR, 20–22 Apr 2017, pp. 1273–1282.

[2] L. Alekszejenkó and T. Dobrowiecki, "SUMO simulations for federated learning in communicating autonomous vehicles: A survey on efficiency and security," *SUMO Conference Proceedings*, vol. 4, pp. 115–129, Jun. 2023. DOI: 10.52825/scp.v4i.221.

[3] P. A. Lopez, M. Behrisch, L. Bieker-Walz, *et al.*, "Microscopic traffic simulation using SUMO," in *The 21st IEEE International Conference on Intelligent Transportation Systems*, IEEE, 2018. [Online]. Available: https://elib.dlr.de/124092/.

[4] F. Caicedo, "The use of space availability information in "parc" systems to reduce search times in parking facilities," *Transportation Research Part C: Emerging Technologies*, vol. 17, no. 1, pp. 56–68, 2009, ISSN: 0968-090X. DOI: 10.1016/j.trc.2008.07.001.

[5] L. Codecá, J. Erdmann, and J. Härri, "A SUMO-based parking management framework for large-scale smart cities simulations," in *2018 IEEE Vehicular Networking Conference (VNC)*, 2018, pp. 1–8. DOI: 10.1109/VNC.2018.8628417.

[6] F. Schaub, Z. Ma, and F. Kargl, "Privacy requirements in vehicular communication systems," in *2009 International Conference on Computational Science and Engineering*, vol. 3, 2009, pp. 139–145. DOI: 10.1109/CSE.2009.135.

[7] 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. DOI: 10.1109/OJCS.2020.2992630.

[8] X. Yin, Y. Zhu, and J. Hu, "A comprehensive survey of privacy-preserving federated learning: A taxonomy, review, and future directions," *ACM Comput. Surv.*, vol. 54, no. 6, Jul. 2021, ISSN: 0360-0300. DOI: 10.1145/3460427.

[9] L. Zhu, Z. Liu, and S. Han, *Deep leakage from gradients*, 2019. arXiv: 1906.08935 [cs.LG].

[10] N. Rodríguez-Barroso, D. Jiménez-López, M. V. Luzón, F. Herrera, and E. Martínez-Cámara, "Survey on federated learning threats: Concepts, taxonomy on attacks and defences, experimental study and challenges," *Information Fusion*, vol. 90, pp. 148–173, 2023, ISSN: 1566-2535. DOI: 10.1016/j.inffus.2022.09.011.

[11] F. Schröter. "Planzeichenverordnung 1990 - PlanzV 90." (in German). (Nov. 2017), [Online]. Available: https://www.dr-frank-schroeter.de/planzv.htm (visited on 01/31/2024).

[12] Eurostat. "Demography of Europe 2023: 2023 interactive edition." (2023), [Online]. Available: https://ec.europa.eu/eurostat/web/interactive-publications/demography-2023 (visited on 02/01/2024).

[13] E. Moradi-Pari, D. Tian, M. Bahramgiri, S. Rajab, and S. Bai, "DSRC versus LTE-V2X: Empirical performance analysis of direct vehicular communication technologies," *IEEE Transactions on Intelligent Transportation Systems*, vol. 24, no. 5, pp. 4889–4903, 2023. DOI: 10.1109/TITS.2023.3247339.

[14] S. Lloyd, "Least squares quantization in PCM," *IEEE Transactions on Information Theory*, vol. 28, no. 2, pp. 129–137, 1982. DOI: 10.1109/TIT.1982.1056489.

[15] L. Codecá and J. Härri, "Monaco SUMO traffic (MoST) scenario: A 3D mobility scenario for cooperative ITS," in *SUMO 2018, SUMO User Conference, Simulating Autonomous and Intermodal Transport Systems, May 14-16, 2018, Berlin, Germany*, Berlin, Germany, May 2018.

[16] J. Day and H. Zimmermann, "The OSI reference model," *Proceedings of the IEEE*, vol. 71, no. 12, pp. 1334–1340, 1983. DOI: 10.1109/PROC.1983.12775.