Vehicular Communications 39 (2023) 100565

Contents lists available at ScienceDirect

Vehicular Communications

journal homepage: www.elsevier.com/locate/vehcom

Privacy pitfalls of releasing in-vehicle network data

András Gazdag [a,*], Szilvia Lestyán [a], Mina Remeli [a], Gergely Ács [a], Tamás Holczer [a], Gergely Biczók [a,b]

[a] CrySyS Lab, Dept. of Networked Systems and Services, Budapest University of Technology and Economics, Hungary
[b] ELKH-BME Information Systems Research Group, Hungary

ARTICLE INFO

Article history:
Received 29 September 2022
Received in revised form 10 November 2022
Accepted 1 December 2022
Available online 9 December 2022

Keywords:
In-vehicle network data
Privacy attacks
Driver re-identification
Trajectory reconstruction
Anonymization
Differential privacy

ABSTRACT

The ever-increasing volume of vehicular data has enabled different service providers to access and monetize in-vehicle network data of millions of drivers. However, such data often carry personal or even potentially sensitive information, and therefore service providers either need to ask for drivers' consent or anonymize such data in order to comply with data protection regulations. In this paper, we show that both fine-grained consent control as well as the adequate anonymization of in-network vehicular data are very challenging. First, by exploiting that in-vehicle sensor measurements are inherently interdependent, we are able to effectively i) re-identify a driver even from the raw, unprocessed CAN data with 97% accuracy, and ii) reconstruct the vehicle's complete location trajectory knowing only its speed and steering wheel position. Since such signal interdependencies are hard to identify even for data controllers, drivers' consent will arguably not be informed and hence may become invalid. Second, we show that the non-systematic application of different standard anonymization techniques (e.g., aggregation, suppression, signal distortion) often results in volatile, empirical privacy guarantees to the population as a whole but fails to provide a strong, worst-case privacy guarantee to every single individual. Therefore, we advocate the application of principled privacy models (such as Differential Privacy) to anonymize data with strong worst-case guarantee.

1. Introduction

The volume of data generated and collected by networked information systems is already staggering and ever-increasing. With the advent of data analytics and machine learning, this data inflow enables thousands of data-driven services which make our lives easier everyday. On one hand, the modern vehicle is such a networked information system, generating abundant data both on its in-vehicle network and, when V2X-enabled, towards other vehicles and the smart infrastructure. On the other hand, a prominent service category is, e.g., location-based services which help us navigate efficiently, hail a ride swiftly and cheaply, and get informed on potentially relevant businesses nearby. However, such services do not come for free: more often than not we pay for them with our personal data.

When service-enabling data is personal, the data controller (and data processors involved) must observe regional and national data protection regulations; in Europe this points to the European General Data Protection Regulation or the GDPR [1]. While we do not intend to introduce GDPR fully in this paper, it is essential to outline three of its concepts to facilitate comprehension. First, GDPR demands legal basis for data controllers to use personal data: the primary way to achieve this is getting the *informed consent* of the data subject. Second, the term "personal data" is defined as "any information relating to an identified or identifiable natural person; an identifiable natural person is one who can be identified, directly or indirectly." *Singling out* is a fundamental method explicitly mentioned to identify a person in data; only data that do not allow singling out the record of *any* individual may be exempt from the GDPR [2]. Third, certain personal data is deemed *sensitive*, and thus enjoy even stronger protection: personal data revealing racial or ethnic origin, political opinions, religious or philosophical beliefs; trade-union membership; genetic data, biometric data processed solely to identify a human being; health-related data; and data concerning a person's sex life or sexual orientation.

Vehicular data often *directly* carries personal or even potentially sensitive information such as location data, heart rate, or driver fatigue among others. Indeed, four spatio-temporal data points

* Corresponding author.
E-mail addresses: agazdag@crysys.hu (A. Gazdag), lestyan@crysys.hu (S. Lestyán), mincsek@gmail.com (M. Remeli), acs@crysys.hu (G. Ács), holczer@crysys.hu (T. Holczer), biczok@crysys.hu (G. Biczók).

https://doi.org/10.1016/j.vehcom.2022.100565

of an individual's daily trajectory identify 95% of all individuals uniquely [3]; this facilitates singling out. Also, as evidenced by the publicly released NYC taxi dataset, individual trips might allow the inference of the driver's (or the passenger's) religious beliefs or health status[1]; these are sensitive personal attributes. Moreover, the Strava incident[2] showed us that location data create patterns on the aggregate that can still give away an individual's personal information.

Meanwhile, data collection through the in-vehicle network, notably the CAN bus, is ongoing by OEMs (Original Equipment Manufacturers) for maintenance, recall, and, increasingly, monetization purposes. Therefore, even third-party service providers can get access to that data; vehicle data hubs collect and standardize such data and sell it for applications including insurance,[3] traffic management, electric vehicle infrastructure planning, fleet management, advertising, mapping, city planning, and location intelligence.[4] The vehicle data market is predicted to be worth between 300 billion and 800 billion USD by 2030.[5] However, if CAN data i) reveals the trajectory and location of the vehicle and its passengers and/or ii) enables singling out a specific driver, it should be treated as personal data; in this case the GDPR should be in full effect. Adding fuel to the fire, CAN data is often collected without the informed consent of the owner/driver and without even the alternative of opting out. Such a hypothetical carries great weight: for handling private data, GDPR defines strict requirements for data controllers and processors (reasonable technical and organizational security safeguards, Art. 24 and 32 GDPR, in some cases data protection impact assessment, Art. 35 GDPR) and rights for data subjects (right to access, right to erasure, right to object, as per Art. 12-23 GDPR). Satisfying these requirements carry a substantial cost and might even perturb the century-old process of how cars are sold. Moreover, not meeting these requirements can result in hefty fines, evidenced by much publicized cases in other domains.[6]

To avoid potential repercussions of data breaches and legal non-compliance, data controllers and processors either apply anonymization, or ask drivers' consent in order to process their data. However, since vehicle data are composed of the fine-grained measurements of a multitude of vehicular sensors, they are inherently high-dimensional, potentially unique to a driver, and therefore challenging to anonymize without significant utility loss. Standard pseudonymization techniques that remove only direct identifiers but keep the sensor measurements intact generally do not work, and vehicular data are no exception.[7] Moreover, the "ad-hoc" application of different standard anonymization techniques, such as the aggregation or removal of apparently sensitive measurements, often results in volatile, empirical privacy guarantees to the population as a whole but fails to provide a strong, worst-case privacy guarantee to every single individual; a specific requirement of privacy regulations. Indeed, as the measurements of different sensors are strongly correlated, we demonstrate later that removing some apparently sensitive measurements (e.g., GPS coordinates) may still allow their inference from the data of other "proxy" sensors (e.g., steering wheel angle and speed), or the identification of the driver from the ensemble of all remaining sensors. This not only makes anonymization very difficult, but it also stifles fine-grained consent control which provides a common legal basis to process personal data if anonymization is not viable. However, data controllers (can) hardly explain the potential privacy implications of sharing such interdependent attributes which means that a driver's consent will arguably not be *informed* and hence becomes invalid.

In this paper, we show exactly that indeed, analyzing CAN logs and using the combination of different sensor measurements make it possible to single out drivers and infer the trajectory of the vehicle even if direct identifiers or precise GPS locations cannot be accessed. Specifically, our contribution is fourfold. First, we reconstruct both short (microtracking) and long (macrotracking) driving routes (including destination) only from the speed, steering wheel position, and the starting location of the vehicle with high accuracy. Our approach is lightweight and provides larger reconstruction accuracy even for longer trajectories (above 1000 meters) than what has been reported before. Second, we re-identify the driver of a vehicle from the contents of recorded CAN logs, either via reconstructing signals or even using only raw data, which has not been considered before. We achieve an average re-identification accuracy of 97% which is comparable to state-of-the-art solutions relying on manually extracted signals. Third, we demonstrate that intuitive but ad hoc de-identification methods providing empirical average-case privacy guarantees cannot be relied on to transform CAN logs into anonymous data exempt from the GDPR [1], while still preserving meaningful utility. Last, we show how differential privacy, providing formal, worst-case privacy guarantees to each individual irrespective of *a priori* knowledge, may be utilized to release aggregated data implementing a viable operating point in the privacy-utility trade-off curve if the data to be released are adequately large.

The rest of this paper is organized as follows. Section 2 provides an overview on related research works. Section 3.1 describes the CAN protocol, its message format and our efforts to collect adequate datasets. Section 3.2 introduces our method to extract meaningful signals from CAN messages, thus enabling the rest of our experiments. Section 4 describes out attacker models. Section 5 shows how we can reconstruct both the short- and long-term trajectory of a vehicle based on its CAN log. Section 6 explains our machine learning model that is able to re-identify drivers from CAN logs with high accuracy. Section 7 investigates multiple naive signal processing and statistical methods meant to provide defense against tracking and re-identification efforts, shows that these are ineffective for CAN data, and advocates for using the differential privacy framework as a viable solution. At last, Section 8 concludes our paper.

2. Related work

2.1. Driver re-identification from CAN data

Driver characterization based on CAN data has gathered significant research interest from both the automotive and the data privacy domain. The common trait in these works is the presumed familiarity with the CAN protocol stack and the manufacturer-specific communication matrix that maps raw CAN messages to sensible sensor signal readings. The latter information is not public; it is usually attained in the framework of some research cooperation with the respective OEM.

Miyajima et al. [4] investigated driver characteristics when following another vehicle, where pedal operation patterns were modeled using speech recognition methods. Sensor signals were collected in both a driving simulator and a real vehicle. Using car-

[1] https://fpf.org/blog/implications-of-broad-data-collection-by-the-nyc-taxi-limousine-commission/.
[2] https://www.theguardian.com/world/2018/jan/28/fitness-tracking-app-gives-away-location-of-secret-us-army-bases.
[3] https://www.nationwide.com/personal/insurance/auto/discounts/smartride/videos/overview5.
[4] https://themarkup.org/the-breakdown/2022/07/27/who-is-collecting-data-from-your-car.
[5] https://www.capgemini.com/insights/research-library/monetizing-vehicle-data/.
[6] https://www.cnet.com/tech/gdpr-fines-the-biggest-privacy-sanctions-handed-out-so-far/.
[7] https://www.vice.com/en/article/4avagd/car-location-data-not-anonymous-otonomo.

following patterns and spectral features of pedal operation signals authors achieved an identification rate of 89.6% for the simulator (12 drivers). For the field test, by only applying cepstral analysis on pedal signals, the identification rate was down to 76.8% (276 drivers). Fugiglando et al. [5] developed a new methodology for near-real-time classification of driver behavior in uncontrolled environments, where 64 people drove 10 cars for a total of over 2000 driving trips without any type of predetermined driving instructions. Despite their advanced use of unsupervised machine learning techniques, authors conclude that clustering drivers based on their behavior remains a challenging problem.

Hallac et al. [6] discovered that driving maneuvers during turning exhibit personal traits that are promising regarding driver re-identification. Wowo et al. [7] showed that choosing the right sub-maneuver can improve driver identification performance. Using the same dataset from Audi and its affiliates, Fugiglando et al. [8] showed that four behavioral traits, namely braking, turning, speeding and fuel efficiency could characterize drivers adequately well. They provided a (mostly theoretical) methodology to trim the vast CAN dataset, focusing on these traits.

Enev et al. authored a seminal paper [9] which makes use of mostly statistical features as an input for binary (one-vs-one) classification with regard to driving behavior. Driving the same car in a constrained parking lot setting and a longer but fixed route, the authors re-identified their 15 drivers with 100% accuracy. The authors had access to all available sensor signals and their scaling and offset parameters from the manufacturer's documentation. In [10], the authors employed Residual Convolutional Neural Network (RCNN), albeit on manually extracted signals.

In [11], the authors use machine learning models to identify new drivers that are different from the drivers in the training phase. Their solution does not require model retraining in the deployment phase. Zheng et al. [12] show that, besides driving patterns, contextual information (driving condition, vehicle type) also helps re-identification because such contextual features can be unique to a driver. Li et. [13] train models only using data collected from onboard sensors without accessing CAN data and only from the driver of interest. Their proposal is somewhat less accurate than closed-world models which rely on CAN data from all drivers. In [14], similarly to our re-identification model, a convolutional neural network (CNN) is used to identify drivers. The authors manually extract around 30 features from CAN data, arrange them into a matrix, and apply two dimensional CNN on this matrix. By contrast, our approach automatically considers every possible feature and leverage signal correlation only between the most identifying signals. In [15], authors re-identify drivers as well as their genders using different machine learning models. Unlike in our setting, drivers drove the same route and the models are trained on already extracted signals.

In a paper targeted at anomaly detection in in-vehicle networks [16], authors developed a greedy algorithm to split the messages into fields and to classify the fields into categories: constant, multi-value, and counter/sensor. Note that the algorithm does not distinguish between counters and sensor signals, and the semantics of the signals are not interpreted. Thus, their results cannot be directly used for inferring driver behavior. Finally, we emphasize that driver re-identification (and as a matter of fact, also vehicle trajectory reconstruction) is an attack on the driver's privacy; therefore even advanced vehicular security mechanisms such as CAN intrusion detection/prevention systems [17] cannot discover or mitigate them. We refer the reader to Section 7 for potential countermeasures.

2.2. Reconstructing vehicle trajectories

The automotive location privacy problem has been researched for well over a decade. A significant boost to this field was the emergence of vehicle usage-based services such as new insurance constructions. These services offer a lower cost for users who shared their vehicular data while promising user privacy via only collecting driving behavior information and not location data. Unfortunately, Dewri et al. [18] already showed that the collection of vehicle parameters can reveal the driving destination without a GPS signal. In the following years multiple papers showed that various types of vehicular data can be used to reconstruct driving traces partially, or completely. Gao et al. [19] showed that based on a starting position and the speed signal, their elastic pathing algorithm could predict destinations with an error smaller than 500 m for 26% of the cases. This is significantly less accurate than our approach which has an error of 5-100 meters for endpoint reconstruction (see Table 2).

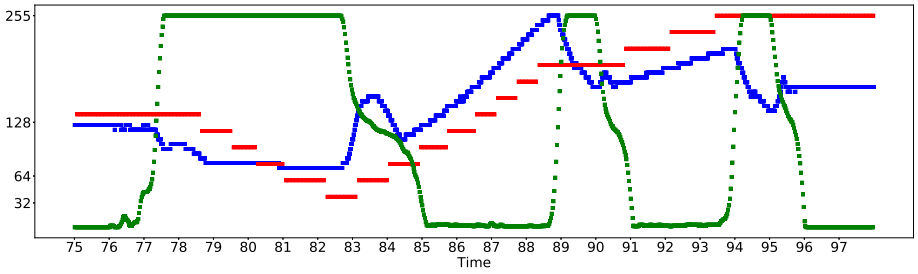
Zhou et al. [20,21] used also only the starting position and the speed signal to reconstruct driving traces. Their approach combines these with additional knowledge about the environment, such as road conditions, real-time traffic, and even traffic regulations, to filter out the potential trace candidates. Results showed that the real route was in the top 10 candidate routes with 60% probability. Kaplun at al. [22] showed that without the detailed speed signal, it was also possible to reconstruct the trajectory with a high probability. They used cornering events, average speed, and total driving time for reconstructing trajectories. Waltereit et al. [23] showed that based on the distance of each driving section and turning directions, it was possible to find the correct trajectory in a large area of a map without knowledge of starting point or destination. Our approach works with similar type of data, however, both the objective and the internal operation of our algorithm differ significantly. The authors in [23] assumed that the signals extracted from CAN messages are accurate and reconstructed route segments separately. However, as we show, signals are inherently erroneous which results in inaccurate reconstruction for longer trajectories.

Pesé et al. proposed the RoCuMa algorithm for route identification based on only the steering wheel position in [24]. The algorithm first creates a road curvature database of the analyzed city, and then tries to identify the curves during the trips. RoCuMa achieved over 70% accuracy in the most suitable regions, but was completely inefficient in others (e.g., Manhattan like grids). The problem RoCuMA tried to solve is similar to ours discussed in Section 5.2, as the steering wheel position can be extracted from the CAN traffic in most cases. Even though the objective is similar to ours, our approach is robust, i.e., its correctness does not significantly depend on the selected region, and also more accurate. Sarker et al. [25] proposed Brake-Based Location Tracking (BBLT). Their approach extracts the break signal values from the CAN bus, which are then used to reconstruct the movement of the vehicle. The extracted signal values are processed in three steps. First, they categorize the signal sub-sequences into different driving maneuvers, then based on the maneuvers they estimate the parameters of the movement, such as the number of intersections or the speed profile. Finally, they search for trajectories on the regional map, that are the closest to the determined parameters. The goodness of a candidate edge is determined by a custom score function. Their evaluation showed that in 89% of the cases the reconstruction was successful.

Finally, the location privacy of a driver can also be violated via smart devices carried on board while driving. E.g., Han et al. [26] described a location inference technique using smartphone accelerometers to successfully locate drivers within a small radius of the true location.

Table 1
Example of CAN messages and the extracted time series. The red, green, and blue time series are obtained by extracting the 1st, 4th and 7th byte of every time-ordered CAN message with ID 0x02c4, respectively.

Timestamp	ID	Len	Data
1481492683.285052	0x0208	0x8	0x00 0x00 0x32 0x00 0x0e 0x32 0xfe 0x3c
1481492674.736055	0x02c4	0x8	0x82 0xc8 0x00 0x0f 0x03 0x00 0x92 0x3c
1481492674.736055	0x02c4	0x8	0x82 0xc9 0x00 0x0f 0x00 0x00 0x92 0x4c
1481492674.736055	0x02c4	0x8	0x82 0xcc 0x00 0x0f 0x08 0x00 0x92 0x5a
1497323915.123844	0x018e	0x8	0x03 0x03 0x00 0x00 0x00 0x00 0x07 0x3f
1497323915.112910	0x00f1	0x6	0x28 0x00 0x00 0x40 0x00 0x00
1481492674.736055	0x02c4	0x8	0x82 0xd2 0x00 0x0f 0x0c 0x00 0x92 0x5d
1481492674.736055	0x02c4	0x8	0x82 0xa1 0x00 0x0f 0xa1 0x00 0x92 0x4d



In summary, prior works use different signals (only speed [18,22,19], speed and traffic information [20,21], only steering wheel position [24], braking [25], acceleration [26]) for reconstruction, while we rely on speed and steering wheel position extracted from CAN messages exclusively. Earlier approaches first partially, or completely reconstruct the trajectory from the signals, then fit the result to the map for correction. By contrast, we always predict the next location from only the previous one using speed and steering wheel position, and immediately correct the prediction with map data, that is, we do not let the error accumulate over successive predictions. As we show later, this approach provides larger reconstruction accuracy even for longer trajectories than what has been reported before. Our technique is simple and efficient, and works even on resource-constrained devices as long as map data can be stored on the device.

3. Background

3.1. CAN: controller area network

The Controller Area Network (CAN) is a bus system providing in-vehicle communications for ECUs and other devices. The first CAN bus protocol was developed in 1986, and it was adopted as an international standard in 1993 (ISO 11898 [27]). A recent car can have anywhere from five to hundreds of ECUs, which are served by several CAN buses. Our point of focus in this paper is mainly the CAN bus serving the drive-train.

The CAN protocol, used on the CAN bus, defines a basic communication format. Each message has an identifier and carries up to 8 data bytes. There are additional fields in the messages like flags for signaling special purpose messages and a checksum to detect transmission errors; these are out of scope for us. There is no standard specification on how signals should be encoded into data fields, therefore each OEM has a custom solution for that. This lack of specification guarantees an added flexibility to the protocol, but also makes it harder for third parties to work with CAN messages. The encoding used for signals is kept as a trade secret by OEMs; however, there are some public projects available online aiming at democratizing access to this information.[8]

CAN is a broadcast protocol, therefore by accessing the network at any available point, all messages can be recorded. The simplest way to connect to the CAN bus is through the On-Board Diagnostic port (OBD). This port was originally developed for maintenance and technical inspection purposes, and it is included in every new car since 1996.

Table 1 shows a simplified picture of a CAN message sequence with a 11-bit identifier, which is the usual format for passenger cars (trucks and buses usually use the extended 29-bit version). This example shows already stripped messages: we do not discuss end-of-frame or check bits and some other parts. In this example, the position of the brake pedal may be carried by the fourth byte of the message with ID `0x02c4`. Therefore, extracting the fourth byte of every consecutive instance of this message type (in the order of their arrival times) one can reconstruct the braking signal (i.e., position of the brake pedal over time, which is shown in green in Table 1). Due to the fact, that this encoding is not publicly available, extracting a signal from recorded CAN messages would require (partly) reverse-engineering the specific CAN communication matrix, which may be illegal under practical circumstances [28].

3.1.1. CAN message structure

- *Timestamp*: Unix timestamp of the message (This is not part of the original message, but the arrival time is necessary for proper processing of the message, therefore we store it for every message upon capture.)

[8] e.g., https://github.com/commaai/opendbc.

- *ID*: contains the message identifier; lower value indicates higher priority
- *Len*: length of the Data field in bytes (0 to 8 bytes)
- *Data*: actual data values; multiple encoded signals per message are typical

3.1.2. Data collection

As CAN data logs are not widely available, we conducted a measurement campaign. For this, we built the hardware and developed the accompanying software of a CAN logging device. The device connects to the CAN bus through the OBD port, and captures all CAN messages, owing to the broadcast nature of communication. Some OEMs limit access to the CAN bus only to diagnostics messages; in those cases, message capture is not possible through OBD. In this paper, we captured CAN traffic in multiple different scenarios to cover a wide range of driving environments.

The dataset used for trajectory reconstruction (both micro- and macrotracking, see 5) contains CAN traffic logs captured from a single mid-class vehicle on different routes. We captured short traces (<100 m) as well as longer traces (1000-2500 m) to test our algorithms in multiple scenarios. In some traces (Fig. 4), we also deliberately drove with rapidly changing steering movements to test the robustness of our algorithms.

For the driver re-identification measurements, we used the dataset from our earlier work [29]. These CAN logs contain the driving logs of 33 drivers from a similar mid-class vehicle: 11 people were between the age of 20-30, 8 of 25-30, 7 of 35-40 and 7 above 40; there were 5 women and 28 men; 12 with little experience (less than 7000 km per year on average or novice driver), 11 with average experience (8-20000 km per year), and 10 with above average experience (more than 20000 km per year). Most routes were driven inside or close to Budapest, Hungary; approximately 15-20% was recorded on a motorway. Drivers were free to choose their way, but still conforming to three practical requirements: (1) record at least 15 minutes of driving in total, (2) do not record data when driving up and down hills, (3) do not record data in extremely heavy traffic (short runs and idling). All drivers drove between 10am and 4pm on weekdays, and the average trace length is 29.81 minutes with a standard deviation of 13.48; the shortest trace is 17.71 minutes long, whereas the longest trace is 85 minutes long.

3.2. Signal extraction

The adversary has to know the higher layer protocols of CAN in order to extract meaningful sensor readings. Since such message and message flow specifications (above the data link layer) are usually proprietary and closely guarded industrial secrets, such adversarial background knowledge might not be reasonable. In this case, the research question changes: is it possible for the adversary to re-identify drivers based on raw CAN data without the knowledge of protocols above the data link layer?

It has been shown in [9] that using only a single signal (the brake pedal) drivers can be re-identified even with 87.33% accuracy, while 100% accuracy can be reached using more than one signal. This means that the adversary does not need to reverse engineer all available signals on the CAN bus, only those which enable the differentiation between drivers. There have been several works [30], [31], [32], [33], [34] on identifying signals and signal boundaries with high efficiency. However, the adversary does not need such sophisticated methods in order to extract a few descriptive signals. Indeed, relying exclusively on the intuition that the physical phenomenon represented by the signal has identical statistical features across different cars, it is possible to identify the same signal in all cars using the same classifier [35]. For example, the accelerator, brake and clutch pedal positions, as well as velocity and engine revolution (RPM) signals can be identified manually in one vehicle that the adversary can access using simple physical observations and expected cross-signal correlations, then the identified signals can be used to train a random forest model in order to more efficiently and accurately extract the same signals in any other, perhaps different type of target vehicle [35].

More specifically, suppose that the adversary can access and use a (test) vehicle and aims to extract signals in the captured CAN log of another (target) vehicle which in turn cannot be used. The signal (or time-series) of velocity can be identified in the test vehicle by matching every candidate series of raw bytes with the real time-series of the velocity (e.g., measured manually during a test drive), using the dynamic time warp (DTW) algorithm [36]. Since brake and accelerator pedal positions are usually not pressed at the same time, they can also be identified by searching for a pair of signals that are exclusive to each other. Furthermore, RPM drops and then suddenly rises upon gear change during acceleration, hence it can also be found by its matching with velocity. Clutch pedal position can also be extracted by its cross-correlation with the already known velocity and RPM signals; when the clutch is released, there is a slight slip around the middle position of the pedal indicating that shafts are starting to connect. Hence, the adversary can search for a pair of signals with one of them having a sharp spike (RPM) and the other a small plateau (slipping clutch) around the same time. Table 1 illustrates these signal characteristics.

Finally, the adversary can train a classifier to learn such distinguishing characteristics of the extracted signals which tend to be universal in most vehicles, and therefore it can recognize the same signals in the captured CAN log of any target vehicle. In [35], 11 statistical features were derived from the signals for this purpose, and finally the accelerator pedal position, the velocity and the RPM signals were identified in 7 different vehicles with a single classifier trained on these features.

4. Adversary model

We consider two adversary models. The first adversary aims to reconstruct the trajectory of a specific individual's vehicle from its CAN data as accurately as possible. The adversary may know the identity of this driver and can only access the vehicle's CAN data, except its GPS trajectory, as well as the geographical map of the region R (e.g., a bounding box of the entire trajectory). This is the only background information the adversary has about the driver's whereabouts. In particular, the GPS location is not present in the CAN data explicitly but can only be inferred indirectly from other signals extracted from the CAN log (see Section 3.2) such as speed, steering wheel position, as well as the start of the trajectory which is presumably observable by the adversary. These are not far-fetched assumptions since the majority of available cars nowadays still do not have built-in GPS receiver, or if they do, it is unlikely that potentially sensitive location information is shared with third parties. The adversary aims to localize the trajectory within R with as little error as possible, which is measured as an average distance between the reconstructed and the original trajectory. Although larger error implies stronger privacy preservation in general, more precise assessment of the potential privacy risks is difficult due to the varying contextual factors and therefore is beyond the scope of this paper. For example, an error within a few hundred meters may not be enough to decide if the driver visited a hospital or a church in a city, but is sufficient to infer which village it traversed in the countryside.

Given a dataset D holding the CAN data of multiple individuals, the second adversary aims to single out the record of a specific (target) individual. Each record is composed of the raw CAN data

captured from the CAN bus of the driver's vehicle without the driver's real identity (i.e., the data is "pseudonymous"). The length of such a capture may span from a few seconds to a few minutes. As opposed to the first adversary, this adversary may not be able to extract any meaningful signal from any record. In order to localize the individual's record in D, the adversary can access some other CAN data of this individual, which is different from the driver's record in D (e.g., it is produced during a different trip), and also some CAN data of other drivers. The adversary attempts to learn the most distinguishing features of the target driver from this background knowledge, and then to identify the record in D with these features. Although larger identification accuracy suggests larger privacy risks, similarly to trajectory reconstruction and as per GDPR, a more precise assessment of privacy risks would require the consideration of other contextual factors which are not discussed in this paper.

5. Trajectory reconstruction

In this section we show that by releasing CAN data, vehicles and thus drivers become traceable. The tracing of the vehicle is achieved in two steps. First, we show how a vehicle can be traced accurately over short distances based exclusively on CAN messages. We refer to this concept as microtracking.[9] Second, we show how to extend tracing for longer trips using additional, publicly available information. We refer to this second problem as macrotracking.[10] Recall that positioning data (GPS) is not included in the CAN traffic.

5.1. Microtracking

The objective of microtracking is to reconstruct the trajectory of a given vehicle over a short distance (10-100 meters). If the initial position is of the vehicle is known, the next position at any time can always be predicted from the previous one using the speed and the direction of the movement. Fortunately, these values can be computed solely from the signals of speed and steering wheel position which are directly accessible in the vehicle's CAN messages transmitted and recorded during the trip.

First, the speed and the steering wheel position signals are extracted from the CAN message stream. These can be relatively easily found even for an unknown vehicle based on the method described in Section 3.2. Next, the extracted signals together with the timestamps of the recorded CAN messages are used to calculate small delta movements of the vehicle. If the vehicle is going along a straight line, then the calculation is straightforward (distance = velocity × elapsed time). If the steering wheel is not in the central position, then the radius of the path segment can be calculated based on the geometry of the vehicle as illustrated in Fig. 1. In particular, the radius R can be calculated from the sine of the angle β of the wheels as $R = L/\sin(\beta)$, where L denotes the distance between the two axles. The wheel position is inferred from the steering wheel position, which is extracted from the CAN messages, through measurements. Finally, the small delta movements are aggregated sequentially to uncover the trajectory of the vehicle. We validated our method successfully in different short but characteristic test scenarios like lane change or obstacle avoidance.

In general, this type of microtracking is reasonably accurate for short trajectories, but it can be quite inaccurate for longer trajectories as the small approximation errors stemming from the noisy and often biased sensor measurements can add up over a longer trip.

[9] https://github.com/CrySyS/MicroTracking.
[10] https://github.com/CrySyS/MacroTracking.

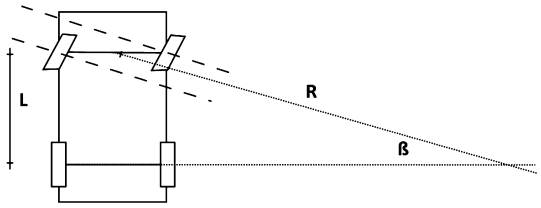


Fig. 1. Computation of the radius as $R = L/\sin(\beta)$, where L and β denote the vehicle length and wheel position, respectively.

5.2. Macrotracking

Next, we describe the reconstruction of the movement of a vehicle from CAN traffic captures over longer trips (>100 meters). To achieve this goal, we use some auxiliary information in order to mitigate the problem of error accumulation. As for microtracking, the speed and steering wheel angle values extracted from the CAN messages are required for the reconstruction. Additionally, the starting position and the initial heading are also a prerequisite for our algorithm. Provided with these input data, we show that the trajectory of a vehicle can be effectively reconstructed revealing the destination of the drive, which constitutes a privacy breach with respect to the driver. This implies that CAN logs have to be handled or processed carefully to avoid this privacy issue and comply with data protection regulations.

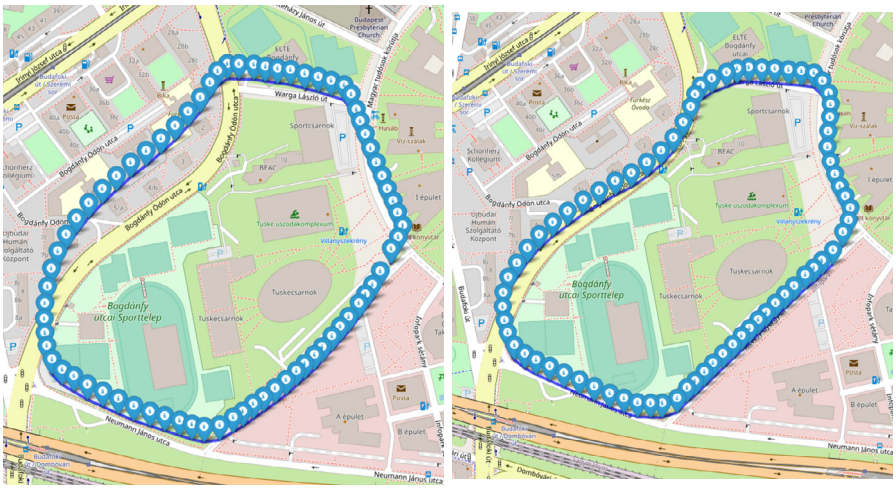
5.2.1. Cumulative errors in microtracking

Our microtracking can be considered as a dead reckoning process which estimates the vehicle position from the previous one using inherently erroneous sensor measurements. Indeed, these measurements are (1) noisy (e.g., steering wheel position, wheel speed measurements are not always accurate due to random environmental factors such as wheel slippage or surface irregularities), (2) potentially biased (e.g., deliberately higher velocity is reported for safety reasons), or (3) some parameters of the vehicle may not be known exactly (e.g., axle distance). Even if these errors result in a small deviation relatively to the previous position, they accumulate over successive predictions making the recovery of longer trajectories very inaccurate. To mitigate this error accumulation, we first performed a thorough calibration by multiplying the extracted values of speed and steering wheel position by a correction factor so that they match the manually measured ground truth values. Although this improves the results, the problem of error accumulation still persists, which is also illustrated by Fig. 2a; some parts of the trajectory are off the road and still far from the original path. In fact, as long as location is predicted only from inaccurate internal parameters and measurements of the vehicle, it remains subject to cumulative errors.

In engineering practice, this problem is generally solved by periodically performing external measurements to correct the prediction and thus reduce the accumulated errors. Similar solutions are used, e.g., for the guidance of spacecraft systems [37]. Next we show how map data can be used to perform periodical measurements to improve the prediction of our model.

5.2.2. Measurements based on map data

The state of a vehicle at any time is characterized by its heading and position. Following our prediction model in microtracking, the state is predicted only from potentially inaccurate internal parameters (axle distance) and sensor readings (speed and steering wheel position), referred to as *model-based prediction* in the sequel. However, using a map allows us to regularly correct the predicted state of the vehicle, i.e., reducing the accumulated errors. In particular, if the vehicle should always be on a road, then a predicted



(a) C1 reconstruction without map (b) C1 reconstruction with map

Fig. 2. C1 test trajectories. Map-corrected trajectories (right) follow the roads more faithfully than only model-based trajectories produced by microtracking (left).

position which is off the road can be corrected. Similarly, the direction of movement should also be aligned with the road, allowing corrections for the heading as well.

Starting from the position predicted by our microtracking model only from internal measurements as described in Section 5.1, we calculate a new position on the map by projecting this model-predicted position on the nearest road segment. This gives us a new, corrected position and heading value aligned with the road that we use in our improved algorithm as a precise external measurement. Along roads without intersections, this approach works smoothly and prevents error accumulation. However, near intersections, it might be unclear which the correct road segment is for the projection. To address this duality in our algorithm, we use a simple linear estimation of the next state as properly weighing both the model-predicted and map-corrected states based on the nearby conditions: the model-predicted states have larger weight near intersections, while map-corrected states have larger weight away from the intersections.

More specifically, given the model-predicted and map-corrected position and heading values, denoted by $state_{model} = [\mathrm{pos}_{model}, \mathrm{head}_{model}]$ and $state_{map} = [\mathrm{pos}_{map}, \mathrm{head}_{map}]$, respectively, the next state of the vehicle is computed as their linear combination: $w \cdot state_{map} + (1 - w) \cdot state_{model}$, where $w = \min(d_{model}, \gamma)/\gamma$ is the map weight, and d_{model} denotes the distance from pos_{model} to the nearest intersection. In other words, if the model-predicted position is within a distance of γ to the nearest intersection, then its weight is inversely proportional to the distance, otherwise mainly map-corrected position and heading are used for prediction.

Our linear estimation is a simple instantiation of the Kalman filter [38], which is a powerful tool to estimate past, present or future states of a system with uncertainties. It needs to have a dynamic model of the target system and multiple sequential measurements to form its estimations. A weighted average is calculated over the output of the dynamic model and measurement results, taking into account the uncertainty of said measurement. In our algorithm, the dynamic model is the movement reconstruction of the vehicle in microtracking when only internal parameters and sensor readings are utilized. The measurements are the projected positions on the map, and the uncertainty of the measurement can be approximated with the distance to the closest intersection.

The pseudo code to reconstruct the movement of a vehicle is presented in Algorithm 1. First, the next state is always predicted from the previous state with model-based prediction as in microtracking (Line 5-8), and then map-based correction (Line 9-14) is only performed if the distance from the last correction is sufficiently large (Line 10 in Algorithm 1). Executing map-based correction after capturing every single CAN message with a speed or steering wheel position value is unnecessary and would prolong reconstruction significantly.

The result of a successful reconstruction of our C1 test is shown in Fig. 2b. Although a slight deviation from the road can be observed near intersections due to model-based prediction, these errors are corrected and hence do not accumulate as moving away from the intersection due to the increasing weight of map-based correction (γ is set to 150 meters).

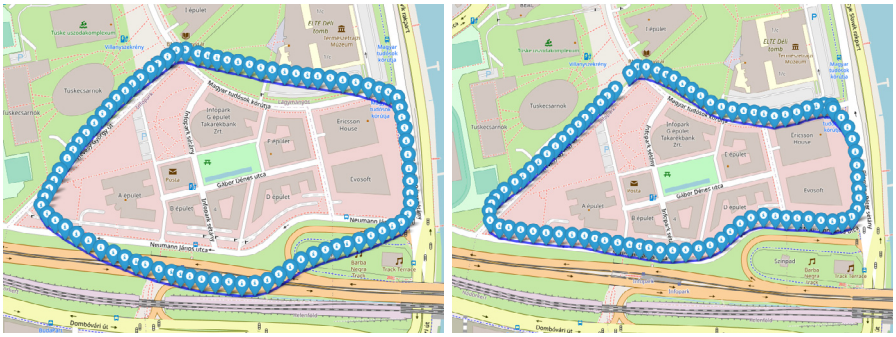
Algorithm 1: Macrotracking for CAN logs.

Input: starting position and heading value, CAN log
Output: Reconstructed trajectory $\mathbb{T}$

```
1  initialize current state to starting position and heading;
2  load data from CAN log;
3  filter relevant messages;
4  while there is message to process do
5      Model-based prediction:
6          extract speed and steering wheel position from messages;
7          compute heading from axle distance and steering wheel
             position;
8          calculate next state from current state using heading and
             speed;
9      Map-based correction:
10         if distance from last correction > minimum required then
11             find nearest road segment on map;
12             project current position and heading to selected road
                 segment;
13             update map weight w based on distance from closest
                 intersection;
14             update next state using the projected state with map
                 weight w;
15     append next state to reconstructed trajectory T;
16     update current state to next state;
```

Table 2
Summary of macrotracking test cases.

Test case	Average trajectory reconstruction error (meter)	Std. deviation of error (meter)	Endpoint reconstruction error (meter)	Total distance travelled (meter)	Number of decision points in map
C1 without map (Fig. 2a)	30.2	25.13	9.3	2025.17	20
C1 with map (Fig. 2b)	9.37	8.99	4.2	2039.11	20
C2 without map (Fig. 3a)	39.37	34.02	35.58	2139.03	18
C2 with map (Fig. 3b)	9.13	8.74	41.12	2158.48	18
C3 without map (Fig. 4a)	55.04	36.07	82.17	1751.07	19
C3 with map (Fig. 4b)	7.45	6.05	6.05	1817.81	19



(a) C2 reconstruction without map (b) C2 reconstruction with map

Fig. 3. C2 test trajectories.

We built our algorithm on OpenStreetMap,[11] which we accessed using the OSMnx library [39]. This service allowed us to get access to precise position and shape information of road segments. We projected the map and all coordinate data in OSMnx from the World Geodetic System 1984 (WGS84)[12] (used in maps and GPS systems) to a Spherical Mercator projection coordinate system (EPSG:3857)[13] to be able to perform more precise distance calculations in 2D.

5.2.3. Validation

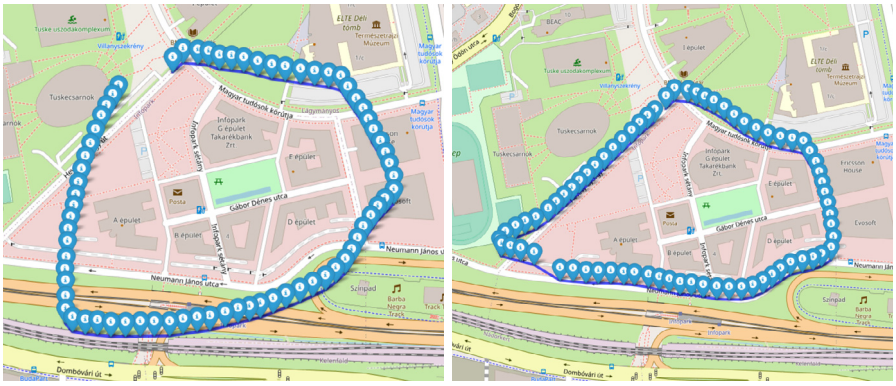
The accuracy of our algorithm depends on the correctness of model-based prediction and the density of the road network. On one hand, areas with many intersections do not allow the map based corrections to improve the model prediction as much, therefore the reconstruction error will dominate over longer distances. On the other hand, if the trajectory of the drive follows long sections without intersections, our algorithm will hardly suffer from any errors.

We defined two metrics to evaluate the accuracy of our algorithm as follows:

1. *Endpoint reconstruction error:* we measure the distance (in meters) between the actual endpoint of the movement and the destination predicted by the algorithm.
2. *Average trajectory reconstruction error:* along the actual movement of the vehicle (the ground truth trajectory) we calculate test points every 15 meters. At each test point we find the closest point in the reconstructed trajectory and measure the distance to the test point. Finally, we calculate an average of the measured distances.

Table 2 shows data about our test cases. In these tests, we drove along different circular trajectories to be able to visually show the result of our first metric as well. We also counted the number of decision points (intersections), where the algorithm had to make a correct decision. In the C3 test case, we drove with frequent movements of the steering wheel even on straight road segments to make the reconstruction harder. The table shows that our algorithm was able to reconstruct the trajectories with only small errors. Furthermore, the corresponding Figs. 2-4 show that the re-

[11] https://www.openstreetmap.org.
[12] https://epsg.io/4326.
[13] https://epsg.io/3857.



(a) C3 reconstruction without map (b) C3 reconstruction with map

Fig. 4. C3 test trajectories.

constructed trajectories follow along the roads actually used, with small deviations only around intersections.

5.2.4. Privacy implications for CAN logs

In this section we showed that it is possible to reconstruct the movement of a vehicle for both short and long distances from CAN messages. This can be a valuable forensics tool, e.g., in case of accident reconstruction; but it can also cause a privacy breach if CAN logs are not handled with proper care and/or they fall into the wrong hands. If the starting position and heading is known, then our algorithm can effectively reconstruct a vehicle' trajectory, revealing private information about the driver. Given that personal location information is categorized as sensitive data in the GDPR, data controllers, whether OEMs or third-party companies, managing such data have to comply with strict requirements. This comes with added cost and operation complexity; yet, non-compliance could result in prohibitively large fines.

6. Driver re-identification

Given a set of already extracted signals from the CAN messages (see Section 3.2), a supervised machine learning model (classifier) can be trained on these signals in order to distinguish every driver. For example, in [35], the accelerator pedal position, the velocity and the RPM signals were extracted and used to re-identify drivers with the average precision of 77%, and reached a top precision of 87% [35]. As described in Section 3.2, most existing driver identification techniques rely on known signals already extracted from CAN data. However, unless the specification of the CAN protocol is known, the adversary needs to reverse engineer the CAN protocol for this purpose, which is often considered to be illegal [28] and tedious.

Fortunately, there is no need to reverse engineer the CAN protocol to re-identify drivers. Indeed, as we show next, drivers can be re-identified directly from CAN messages without interpreting their content utilizing artificial deep neural networks. As demonstrated in the last decade, such models are capable of automatically learning higher-level features from the raw, uninterpreted input data with limited or no manual feature engineering. Therefore, even a weaker adversary, who does not know the potentially proprietary CAN protocol, could distinguish a driver from the rest with large enough accuracy.

We present a machine learning model from [29] specifically designed to distinguish drivers only from CAN data *without* any explicit signal extraction. In [29], the model takes the sequences of bytes extracted from identical positions of CAN messages with the same type as an input time series, and outputs which driver these time series belong to. Importantly, the semantic meaning of each byte and, in particular, the interdependence of bytes within a CAN message (i.e., which bytes belong to the same signal) are unknown to the adversary. In our current work, we use exactly the same model for re-identification, but improve on data pre-processing.

Table 3
Re-identification accuracy: One-vs-all with sample length of 120 sec.

	Sample len	One-vs-all Accuracy			
		Mean	Std. dev.	Min	Max
Original approach (w/o [40])	120 s	85%	0.025	0.47	1.00
Our approach (with [40])	120 s	97%	0.07	**0.81**	1.00

6.1. Overview

In the original paper [29], every byte from a CAN message is extracted, then the same bytes of every CAN message with identical message ID are concatenated (such as in Table 1) to build the time series from which the ones with the most predictive power are identified. Although results show that even such a simple approach results in high re-identification accuracy (see first row in Table 3), this is an overly naive approach that does not consider real signal boundaries (that often differs from byte boundaries) nor the endianness of the byte (that can differ between CAN messages). We have improved on this approach and identified potential signal boundaries by applying [40] to find the correct bit positions of a signal in a message considering endianness.[14] We stress that this technique [40] only identifies signal boundaries, the semantic meaning of the extracted time-series is still unknown and not required for our re-identification model.

The model is a convolution-based neural network, where features from individual time series are learnt automatically by convolutional layers, which are then combined in a mixture model in order to perform the final classification. This mixture model has two main components: (1) the *Individual time series (ITS) model* is specialized to the classification of a single time series. One ITS model per time series is built, hence K different ITS models are obtained, where K is the number of all time series. (2) The *Mixture model* is combined with the already trained ITS models into a single fully connected decision/output layer which provides the final classification result. ITS models are trained individually and sepa-

[14] This method is considered state-of-the-art for signal extraction. Code published by the authors is available at https://github.com/brent-stone/CAN_Reverse_Engineering.

rately from the mixture model. The model can be seen on Fig. 11 in Appendix B.

6.2. Pre-processing

A signal is first divided into equally-sized segments, from here on referred to as samples. Given a single sample as input, the model attempts to classify it into its correct class (driver). A sliding window is used over the whole time series to create samples. Specifically, let $T = (t_1, t_2, \ldots, t_{|T|})$ denote a time series, and let $T_{i:j} = (t_i, t_{i+1}, \ldots, t_j)$ be a window of T between positions i and j $(1 \le i \le j \le |T|)$. Then, the set of all n samples created from T is given by $\{T_{1,n}, T_{1+\ell,n+\ell}, \ldots, T_{|T|-\ell+1,|T|}\}$, that is, all consecutive windows shifted by ℓ time slots.

6.3. Individual time series model (ITS)

As the ITS model is applied on each segment of the sample independently, it can model local features of a segment, the sequential information along the sequence of segments and any global features of the whole sample. In particular, a single CNN model is used to extract features from every segment s_i. The sequence of these representations are further processed by a Long-Short Term Memory model (LSTM), which computes time-dependent features by transforming the input sequence into another sequence $h_1, h_2, \ldots, h_k$ composed of the hidden states of the LSTM. Specifically, for input segment s_i, $h_i = \mathrm{LSTM}(\mathrm{CNN}(s_i), h_{i-1})$. The sequence of $h_1, h_2, \ldots, h_k$ is fed into an attention layer in order to select those parts of the sequence which represent distinctive driver features. In particular, a sample may contain only a few segments with clear distinctive features of the drivers such as a unique pattern of speeding, steering, or braking, while the rest may lack any unique driving pattern and hence bearing less importance in distinguishing drivers.

6.4. Mixture model

Finally the ITS models are combined into a single Mixture model by connecting all ITS models to the same fully-connected neural layer, called mixture layer, whose output provides the final prediction. Only the top-K best performing ITS models are used in the mixture. This mixture model is a sort of *committee machine*, where multiple neural networks, called as experts, are combined to provide the final classification. Indeed, each ITS model is an expert which is specialized to extract a high-level representation of a single time-series. The mixture model is trained to distinguish one driver from all the rest (1-vs-all), and hence it is output is binary.

6.5. Results

Since a mixture model distinguishes one specific driver from all the rest, we train one model per driver, and report different statistics of accuracy over all models in Table 3. To measure performance on the test set, the (test) accuracy is computed as the number of all correct classifications divided by all classifications. The average re-identification accuracy over all the 33 drivers is 85% for the original approach [29] (first row in Table 3), when every consecutive one- and two-bytes in all CAN message are blindly fed into the mixture model. However, when potential signals are first extracted by applying [40], re-identification results improved significantly with a 12% increase in average accuracy, and the minimum accuracy (for the driver whose model has the smallest accuracy) grew by an immense margin of 34% (second row in Table 3). Interestingly, we found 255 potential signals using [40], where only 80 were on the exact same position among all drivers and 120 were "close", i.e. only 1-2 bit apart. The results in the second row of Table 3 were computed only on these 80 common signals. This also prevented the model from identifying a driver based on the occurrence of a signal unique to the driver (e.g., windshield-wiper signal if only one driver drove in rainy weather conditions).

Recall that the adversary aims to single out the record (i.e., CAN data) of a specific (target) individual among several others in a "pseudonymous" dataset. To do that, the adversary trains a mixture model to distinguish the CAN data of the target from all the rest (1-vs-all), and applies the trained model on each record of the dataset. Finally, it randomly chooses one from the records that are classified as the target. To compute the success probability of this attack, we need additional performance metrics of the model: test sensitivity TP/P (the ratio of correctly classified positive samples) and test specificity TN/N (the ratio of correctly classified negative samples), where a sample is positive if it belongs to the target, otherwise negative. In case there is one record from the target and $n-1$ from others, the success probability of the singling out attack is approximated as

$$\begin{aligned}&\Pr(\text{Success})\\ &\approx \sum_{k=0}^{n-1} \binom{n-1}{k} \left[\frac{\mathrm{TP}}{\mathrm{P}} \cdot \left(\frac{\mathrm{FP}}{\mathrm{N}}\right)^k \cdot \left(\frac{\mathrm{TN}}{\mathrm{N}}\right)^{n-1-k} \cdot \frac{1}{1+k} \right] =\\ &= \sum_{k=0}^{n-1} \binom{n-1}{k} \Big[\text{Sensitivity} \cdot (1-\text{Specificity})^k \\ &\quad \cdot \text{Specificity}^{n-1-k} \cdot \frac{1}{1+k} \Big] \end{aligned} \tag{1}$$

where TP is the number of true positives, TN is the true negatives, FP is the false positives, FN is the false negatives, $\mathrm{P} = \mathrm{TP} + \mathrm{FN}$ is the number of all positive samples and $\mathrm{N} = \mathrm{TN} + \mathrm{FP}$ is the number of all negative samples.

To see why Eq. (1) holds, recall that $\mathrm{N} = n - 1$ since there is exactly one positive sample (out of n), and the probability that the target is positively classified by the model can be estimated by the sensitivity ratio. Moreover, the model produces exactly k false positives with a probability of $(1 - \text{Specificity})^k$ and $n-1-k$ true negatives with the probability of $\text{Specificity}^{n-1-k}$, finally the probability that the attacker randomly chooses the target among all positively classified samples is $\frac{1}{1+k}$. Furthermore, since each false positive belongs to a different driver, we need to sum over all possible combinations of k drivers from $k = 0$ till $k = n-1$. We calculated the specificity and sensitivity values on the test set and report them in the next section with each defense technique.

7. Defenses

In this section, we describe and evaluate different defense techniques against location reconstruction (macrotracking) and driver re-identification described in Section 5.2 and 6, respectively. First, we consider some well established signal processing techniques including low pass filtering and smoothing which might be tempting to employ by a data controller or processor in order to distort CAN data so that unique features of drivers are no longer recognizable. However, as we show, such techniques fail to provide strong privacy guarantees, or introduce so much distortion to counter our attacks that renders the anonymized data practically useless. As a result, we consider anonymization techniques with more rigorous privacy guarantees in Section 7.4.

7.1. Smoothing

Smoothing is a type of downsampling technique, it is used to smooth short-term fluctuations and highlight longer-term trends

Table 4
Accuracy of one-vs-all re-identification depending on smoothing window size in seconds.

Smoothing window	One-vs-all Accuracy			
	Mean	Std.dev.	Min	Max
0.1	0.97	0.07	0.81	1.00
0.2	0.96	0.03	0.84	1.00
0.4	0.95	0.04	0.79	1.00
0.8	0.92	0.11	0.48	1.00
1.6	0.77	0.17	0.50	1.00
3.2	0.86	0.14	0.46	1.00
6.4	0.71	0.17	0.43	1.00

in the signal (or time-series). It has many variations, the main idea is to shift a moving window of a fixed size through the signal and apply a transformation on each window, then publish the transformed signal. We apply a moving window by passed time and not by data points, i.e. the average of the data points that is in a fixed time frame (called *smoothing window*), whose size in time is given as a parameter w, is calculated and used as a single replacement of all points within the given time frame. As the mean is reported per window, consecutive windows do not overlap. This technique is expected to smooth out local variations of every signal within w seconds. Therefore, as long as such local variations correspond to unique features of a driver, the transformed signal should mitigate re-identification and tracking.

7.1.1. Evaluation

We evaluate our re-identification model on the smoothed data produced with different values of smoothing window size w. The goal is to find a window size where the accuracy of re-identification and macrotracking are sufficiently small but the accuracy of the transformed data is still meaningful.

Driver re-identification We applied smoothing with different window lengths (shown in seconds), then we used the re-identification algorithm from Section 6 on the smoothed signals. Table 4 shows that, by increasing the window size, the re-identification accuracy drops only until 0.71, but the worst case success probability of singling out (the maximum value of Eq. (1) over all drivers) is 100% in all cases, that is, there is always a driver whose record can be singled out. The first line corresponds to the same window size that was used in [29], that is the lowest frequency used by any signal in the CAN bus. The window size is doubled in consecutive lines of the table. At smaller window sizes, the re-identification even improves, which is probably due to the de-noising feature of smoothing. However, increasing the window size further, the average accuracy drops as expected. Importantly, some drivers remain perfectly re-identifiable even with the largest window size. This could be due to some very specific signals which appear only in a particular driver's CAN data (e.g., windshield wiper is used only by a single driver), or a driver could have a unique *global* feature across multiple smoothing windows or multiple signals (e.g., the average speed over the whole trip) which is not removed by smoothing out only local variations of a signal.

Obviously, a data controller/processor cannot account for all potential idiosyncrasies of a driver's CAN data. Smoothing reduces the *mean* accuracy, the *mean* specificity and the *mean* sensitivity of re-identification, but some drivers still remain 100% identifiable which is also shown in Table 4. Moreover, the average re-identification accuracy, specificity and sensitivity only drops to 0.66-0.71 at a window size of 6.4 seconds, which means that only a single datapoint is retained over during 6.4 seconds that is the average of 60-600 measurements (Table 5). This resolution is clearly too coarse-grained for many applications (e.g., for forensics if accidents can happen in less than one second).

Location tracking Fig. 5 shows the effect of smoothing on the model-based trajectory reconstruction without map correction (i.e., microtracking). Although smoothing negatively impacts reconstruction, applying smoothing even with the largest window size still allows relatively accurate reconstruction many parts of the trajectory. Interestingly, due to the special encoding of the steering

Table 5
Worst case success probability of singling out attack (by Eq. (1)), specificity and sensitivity of one-vs-all re-identification depending on smoothing window size in seconds. The length of the smoothing window is in seconds.

Smoothing window	Singling out success	Specificity				Sensitivity			
		Mean	Std.dev.	Min	Max	Mean	Std.dev.	Min	Max
0.1	1.00	0.99	0.02	0.91	1.00	0.96	0.15	0.71	1.00
0.2	1.00	0.98	0.04	0.90	1.00	0.98	0.05	0.66	1.00
0.4	1.00	0.93	0.05	0.79	1.00	0.98	0.07	0.57	1.00
0.8	1.00	0.86	0.18	0.32	1.00	0.95	0.19	0.00	1.00
1.6	1.00	0.81	0.24	0.13	1.00	0.76	0.41	0.00	1.00
3.2	1.00	0.79	0.23	0.07	1.00	0.87	0.28	0.00	1.00
6.4	1.00	0.66	0.28	0.00	1.00	0.69	0.34	0.00	1.00

(a) Smoothing window: 0.201s (b) Smoothing window: 1.608s (c) Smoothing window: 6.4s

Fig. 5. C1 test case macrotracking result on smoothed data without map.

(a) Smoothing window: 0.201s (b) Smoothing window: 1.608s (c) Smoothing window: 6.4s

Fig. 6. C1 test case macrotracking result on smoothed data with map.

Table 6
Effects of smoothing on our location tracking algorithm.

Test case	Smoothing windows size (second)	Average trajectory reconstruction error (meter)	Std. deviation of error (meter)	Endpoint reconstruction error (meter)
C1 (Fig. 6a)	0.201	32.2	26.4	22.76
C1 (Fig. 6b)	1.608	33.51	26.97	33.02
C1 (Fig. 6c)	6.4	38.58	27.97	132.94
C2 (Fig. 13a)	0.201	37.75	28.68	75.74
C2 (Fig. 13b)	1.608	38.92	32.83	76.72
C2 (Fig. 13c)	6.4	42.22	32.47	126.84
C3 (Fig. 15a)	0.201	50.78	32.74	64.71
C3 (Fig. 15b)	1.608	47.53	29.6	66.52
C3 (Fig. 15c)	6.4	20.47	18.15	70.09

wheel value, applying smoothing on this signal results in sharper turns than in the original trace.

Our macrotracking algorithm with map correction is significantly more accurate than only model-based reconstruction and can successfully reconstruct the original traces in all cases (see Figs. 13 vs. 12 for C2 and Figs. 15 vs. 14 for C3 in Appendix C, respectively). In the C1 (Fig. 6) and C2 (Fig. 13) cases, increasing the window size also increases the reconstruction error, but all results can still be considered as successful reconstructions. Although the C3 test case is driven with frequent steering wheel changes (as described in Section 5.2.3), the effect of these changes is reduced by smoothing, and therefore the reconstruction results are actually improved with a larger window size (Fig. 15). Table 6 contains the trajectory reconstruction errors with their standard deviation and the endpoint reconstruction error for all test cases with three different window sizes.

7.2. Low pass filtering

Low pass filtering is a common technique not only to compress signals, but to reduce noise, eliminate aliasing, or attenuate resonances [41] without heavily degrading utility. Moreover, with the growing need for data privacy, low pass filtering has been used for signal anonymization as well [42]. Low-pass filters attenuate or eliminate all signal components above a specified frequency. By deleting these high frequency components, one can get rid of the idiosyncrasies of the signal and end up with the more general parts. For example, by using the brake pedal signal only (this contains the angle the pedal is pushed in by), a driver can be re-identified with an accuracy of 87% [9]. In this signal the high frequency parts represent small changes, and the overall tendency is determined by the lower frequencies (e.g., small frequencies can represent the unevenness of the movement of pedal pushing). For this reason it is hypothesized that the high frequency components of a signal identify a driver. Such components correspond to subtle and small changes in the signal that differ most significantly among drivers, thus by eliminating some of these high frequency components one could also remove a drivers' identifying features in a signal. Unlike smoothing, low pass filtering is expected to provide a finer-grained control over utility loss, and the mean squared error is precisely quantifiable since the transformation is orthonormal and therefore preserves the L_2-norm of the signal.

In this section, we show that low pass filtering still implies mostly unsatisfactory privacy guarantee in practice. Below we explain how we applied a low-pass filter on CAN bus signals with different degrees of utility, then we evaluate the resulting filtered signals both with our re-identification (Section 6) and macrotracking attacks (Section 5.2).

We apply low-pass filtering as follows. First, the signal is transformed to its frequency domain using orthonormal Discrete Cosine Transform (DCT) which has better energy compaction property than other Fourier-related transforms. After DCT transformation, the number of removed high frequency components is determined. In general, the more components are dropped from the signal the lower the utility becomes. The resulting utility is measured by calculating the normalized euclidean distance between the original and the low passed signal, i.e. we delete as many of the highest frequency components as many needed to reach a predefined *error distance* (aka., reconstruction error) from the original signal. As orthonormal DCT preserves the L_2-norm of the original signal, the transformed signal has the same L_2-norm as the original one. For example, in order to have a reconstruction error of 10% at most, the maximum number of the highest frequencies of the transformed signal are removed such that the L_2-norm of the removed components is not greater than the 10% of the total L_2-norm of the whole signal. A naive algorithm would start deleting components one-by-one starting from higher to lower frequencies, instead we apply logarithmic search and calculate the resulting error after each iteration. Once the desired error rate is reached, the filtered signal is transformed back to the time domain and published.

7.2.1. Evaluation

Driver re-identification We applied low-pass filtering with different error rates, then we used the re-identification algorithm from Section 6 on the low-passed signals. The results are depicted in Table 7. The worst case success probability of the singling out attack (calculated by Eq. (1)) is always 100%. Mean accuracy shows the average one-vs-all re-identification accuracy over 33 drivers from our database (see details in Section 3.1). Interestingly, the minimum re-identification accuracy is even improved with moderate low pass filtering producing a reconstruction error of 10%, and there is only a slight drop in mean accuracy and sensitivity, and no change in mean specificity at the higher reconstruction error of 40%. Indeed, low pass filtering is often used as a pre-processing step to de-noise the original signal and hence improve classification. In fact, some signals are so compact in the frequency domain that only a few frequency components are retained after filtering. However, this does not always prevent re-identification: even if we end up only with the lowest frequency (i.e., a constant transformed signal) after filtering, it can still be unique to a single person in the whole dataset. For example, a signal which is unique to the driver will always re-identify the driver no matter how many of its components are dropped. Moreover, the same data peculiarities already described in Section 7.1.1 also hold here. As a result, some drivers remain re-identifiable even with the largest reconstruction error which is also shown in Table 7. The reason we had to stop the evaluation at an error rate of 60% is that several signals could barely (or could not) reach this prescribed error rate without dropping all of its components, and hence we dropped all such signals completely. However, above the error of 60%, almost all signals would be dropped which would yield useless (but perfectly anonymized) data with meaningful utility.

Consequently, low-pass filtering, as a general anonymization technique, does not provide sufficiently strong privacy guarantees for every single driver (Table 8).

Location tracking Fig. 7 shows the result of trajectory reconstruction without map (i.e., only model-based prediction) after low-pass filtering the C1 test case. In comparison with smoothing, low-pass filtering with the chosen parameters distorts the original traces more significantly. The counter-intuitive changes of the turn angles can also be observed here.

Fig. 8 shows that our macrotracking algorithm (i.e., reconstruction with map correction) is capable of reconstructing the original C1 trajectory if the prescribed error rate of low pass filtering is below 40%. Above this value, the algorithm cannot reconstruct the vehicle movement in one of the intersections at least. We also observed similar behaviors in the C2 (Figs. 17 vs. 16) and C3 (Figs. 19 vs. 18) test cases, see Appendix C. The accuracy of the reconstruction for all cases with different amount of low-pass filtering is depicted in Table 9. The reconstruction is successfully prevented at a low pass filtering error of 40%.

Table 7
Accuracy of one-vs-all re-identification with low pass filtering for different error rates. Note that the first line is the baseline.

Error	One-vs-all Accuracy			
	Mean	Std. dev.	Min	Max
0%	0.97	0.07	0.81	**1.00**
10%	0.97	0.03	0.88	**1.00**
40%	0.96	0.06	0.76	**1.00**

7.3. Aggregation

In the previous two subsections we have seen that "ad-hoc" anonymization methods do not provide strong privacy guarantees to every single individual, or if they do, the resulting data accuracy is often unsatisfactory in practice. Next, we show that releasing only aggregated information about CAN datastreams can still result in privacy breaches. This will motivate the application of Differential Privacy, which provides strong, provable privacy guarantees to every single individual.

Oftentimes, in order to deduce meaningful statistics from a vehicle, there is no need to publish raw CAN bus signals. Several weeks or months of data can occupy even several terabytes, where large part of it is unnecessary or redundant. Not to mention that, as shown in Section 6 and Section 7.2, it is hard to anonymize them in this raw, micro-data form, where separate records (time series) are released about each individual. However, such micro-level data reporting is not required by many applications. For example, a fleet management company could be interested in the general state of its vehicles, each possibly driven by multiple individuals, that is, how much some cars have been deteriorated during the past year or so. With this aim the data owner does not have to publish all the raw data of the vehicles, but can precalculate the needed information, such as the overall milometer, how often or how much a vehicle has run over a specified rpm, what its average speed was or how much it has run with a speed more than 130 km/h. A straightforward solution is to answer only aggregated queries over the derived dataset and excluding those answers that are true only for a few drivers. However, it is a common misconception that aggregation per se (performed over several individuals) preserves privacy. In this section we show a systematic approach to reconstruct individual-specific information from CAN bus data only using aggregate queries.

Data reconstruction from aggregates Let D be a dataset with n records, and a record has a private attribute denoted by x_i, such that the vector of all private attributes is $X = (x_1, x_2 \dots x_n)$. The

(a) Filtering: 10% (b) Filtering: 20% (c) Filtering: 40%

Fig. 7. C1 test case macrotracking results on low pass filtered data without map.

(a) Allowed reconstruction error: 10% (b) Allowed reconstruction error: 20% (c) Allowed reconstruction error: 40%

Fig. 8. C1 test case macrotracking results on low pass filtered data with map.

database owner wishes to preserve privacy by only answering aggregate queries concerning the private attribute without revealing any $x_i \in X$. Formally, an aggregate query $q(D)$ specifies a subset of the records $Q \subseteq \{1, \ldots, n\}$ satisfying a certain predicate, and an aggregate function f such as sum. Each record is endowed with some demographic information such as age, sex or driving experience. For example, if x_i denotes the average speed of a driver, Q consists of all male drivers with age 32, and f denotes function SUM, then $q(D) = \sum_{i \in Q} x_i$, which can be specified using an SQL-like syntax as:

```
SELECT SUM(avg_velocity) WHERE AGE = 32 AND SEX = 'Male';
```

In general, $q(D)$ is the result of f applied on subset $\{x_i | i \in Q\} \subseteq X$.

Our aim is to avoid the disclosure of any single private attribute value x_i from a set $\{q_1(D), \ldots, q_n(D)\}$ of query results:

Definition 7.1. (Full Disclosure [43]) A private attribute value $x_i \in X$ is fully disclosed by queries $\{q_1, \ldots, q_n\}$, if x_i can be uniquely determined, i.e., in all possible data sets D with private attribute values X consistent with the answers $\{q_1(D), \ldots, q_n(D)\}$, x_i is the same.

For example, if the query set consists of a single query asking for the sum of the average velocity of all the male drivers in the dataset, but Bob is the only male driver in the fleet, then the answer to this query reveals Bob's average speed. A naive defense would be denying such queries, where the output contains only a single record, however this does not help much. Consider two queries where the first returns the sum of average speeds over all n drivers, and the second returns the sum of average speeds over all $n-1$ female drivers. Even if $n >> 1$, the difference of these two queries uniquely determines Bob's speed. In other words, the linear combination of multiple different queries can potentially reveal a private attribute value x_i.

To make this attack more concrete, suppose that each record is defined as $n_i = (id_i, x_i)$ where id_i is some identifying information also known to the adversary (such as the combination of age, sex and experience in our case), and $x_i \in \mathbb{R}$ is the private attribute (i.e., average speed). If W is the set of all possible identifiers, and Q is specified by the function $\phi : W \to \{0, 1\}$, then a query can be defined as

$$q_i(D) = \sum_{j=1}^{n} \phi_i(id_j) \cdot x_j$$

where $\phi_i(id_j) = 1$ if the ith query covers record j and 0 otherwise, i.e. each query is represented by a binary vector indexing the

Table 8
Sensitivity, Specificity, and worst-case success probability of singling out attack (by Equation (1)) of re-identification with low pass filtering for different error rates (One-vs-all re-identification). Note that the first line is the baseline.

Error	Singling out Success	Sensitivity				Specificity			
		Mean	Std. dev.	Min	Max	Mean	Std. dev.	Min	Max
0%	1.00	0.99	0.02	0.91	1.00	0.96	0.15	0.71	1.00
10%	1.00	0.97	0.03	0.84	1.00	0.96	0.05	0.74	1.00
40%	1.00	0.93	0.13	0.51	1.00	0.96	0.03	0.88	1.00

Table 9
Effects of low pass filtering on our location tracking algorithm.

Test case	Allowed reconstruction error	Average trajectory reconstruction error (meter)	Std. deviation of error (meter)	Endpoint reconstruction error (meter)
C1 (Fig. 8a)	10%	8.63	9.07	8.45
C1 (Fig. 8b)	20%	8.25	7.33	12.9
C1 (Fig. 8c)	40%	47.71	74.23	602.64
C2 (Fig. 17a)	10%	8.71	8.94	11.43
C2 (Fig. 17b)	20%	10.98	11.9	13.37
C2 (Fig. 17c)	40%	175.08	159.73	589.17
C3 (Fig. 19a)	10%	7.01	5.92	9.36
C3 (Fig. 19a)	20%	7.58	5.93	8.16
C3 (Fig. 19a)	40%	207.08	151.12	124.05

records that are covered by the sum query. Therefore, the answers to a set of queries $q_1, \ldots, q_k$ is given by the following matrix-vector product $Q_W \cdot x$:

$$\begin{bmatrix} q_1(D) \\ \vdots \\ q_k(D) \end{bmatrix} = \begin{bmatrix} \phi_1(id_1) & \ldots & \phi_1(id_n) \\ \vdots & \ddots & \vdots \\ \phi_k(id_1) & \ldots & \phi_k(id_n) \end{bmatrix} \begin{bmatrix} x_1 \\ \vdots \\ x_n \end{bmatrix} \quad (2)$$

where Q_W is a binary matrix. In order to check whether any x_i is revealed by the query results $q_1(D), \ldots, q_k(D)$, the adversary tries to solve Equation (2) for any x_i. Since this is a system of linear equations and $x_i \in \mathbb{R}$, this can be done in polynomial time using any LP solver.[15] In case id_i is not unique to record i, i.e. there exists j such that $id_i = id_j$, then matrix Q_w becomes singular and Equation (1) cannot be solved uniquely but only approximated (e.g., with ordinary least square). Similarly, in case f is a non-linear function (e.g., MIN, MAX, MEDIAN) or $x_i \in \mathbb{Z}$, then query auditing is not guaranteed to run in polynomial time but can only be approximated with some heuristics [44].

Results We implemented the above reconstruction attack and tested on our dataset with 33 drivers. After dropping out records with identical identifiers, we obtained 28 drivers each with a unique combination of age, sex, and experience. Then, we generated random queries (i.e., each query covered a random subset of records), and attempted to solve the obtained system of equations given in Equation (2). Only 100 random queries allowed the complete reconstruction of the private attributes values (average speed) of all the 28 drivers.[16]

A possible defense against the above reconstruction attack is query auditing [45]. Here, the data controller keeps track of all answered queries and allows/denies a new query by checking if answering the new query allowed to solve Equation (2) for any x_i. However, query auditing has a few drawbacks. First, even if it runs in polynomial time, it is often not scalable to many queries. Second, non-linear aggregation function, or aggregation on integer-valued private attributes cannot always be audited efficiently but only approximated, as discussed above, without strong guarantees. Finally, query auditing cannot take into account all possible adversarial auxiliary knowledge. For example, the adversary may be able to squeeze the possible ranges of private attribute values which introduces extra (convex) constraints in Equation (2) providing more accurate solutions (e.g., it knows that Bob never drives faster than 60 km/h, or the solution space $\mathbf{x}$ is sparse). A more promising approach is to add random noise to the query results and return these noisy results to the querier. Assuming that all adversarial auxiliary knowledge is known to the data controller, such approach can still be audited by solving a system of inequalities [46], which provides a lower bound on the variance of the added noise that are necessary to avoid full disclosure. This approach initiated the development of Differential Privacy.

7.4. Differential privacy

Differential privacy [47] (DP) ensures that the outcome of any computation on a database is insensitive to the change of a single record. It follows that any information that can be learned from the database with a record can also be learned from the one without that particular record. In our case, DP guarantees that a query is not affected by any single record (driver) beyond the privacy budget measured by ε and δ, which can be computed as follows.

Definition 7.2 (*(ϵ, δ)-**Differential Privacy** [48]*). A privacy mechanism $\mathcal{M}$ gives (ϵ, δ)-Differential Privacy if for any database D_1 and D_2 differing on at most one record and for any subset of outputs $S \subseteq Range(\mathcal{M})$:

$$\Pr[\mathcal{M}(D_1) \in S] \leq e^{\varepsilon} \Pr[\mathcal{M}(D_2) \in S] + \delta$$

Intuitively, a privacy mechanism $\mathcal{M}$ satisfying Definition 7.2 does not release any information that is specific to any single record in dataset D up to ε and δ, where δ is preferably smaller than $1/|D|$ [48]. Importantly, unlike earlier anonymization techniques, this DP provides a worst-case guarantee independently of what the adversary knows; (ϵ, δ) holds for any individual both inside and outside dataset D, no matter what auxiliary knowledge the adversary has.

It follows immediately from Definition 7.2 that $(\epsilon, 0)$-differential privacy composes in a straightforward way: the composition of k $(\epsilon, 0)$-differentially private mechanisms is $(k\epsilon, 0)$-differentially private. For any $\delta > 0$, the k-fold adaptive composition satisfies (ϵ', δ')-DP, where $\epsilon' = 2\epsilon\sqrt{2k\log(1/\delta')}$ [49]. Moreover, any processing of the output of a differentially private mechanism $\mathcal{M}$ does not degrade its privacy guarantee, which is often referred to as the post-processing property of DP [47].

A fundamental concept for achieving differential privacy is the global sensitivity of a function [47]:

Definition 7.3 (*Global L_p-sensitivity [49]*). For any function $f : D \to \mathbb{R}^d$, the L_p-sensitivity of f is $\Delta f = \max_{D_1, D_2} ||f(D_1) - f(D_2)||_p$ for all D_1, D_2 differing in at most one record, where $||\cdot||_p$ denotes the L_p-norm.

There are several techniques to achieve (ε, δ)-DP. For simplicity, we focus on the stronger privacy-preserving case when $\delta = 0$ and use the Laplace Mechanism [49], which adds Laplacian noise to the true output of a function. In particular, for any function $f : D \to \mathbb{R}^n$, the Laplace mechanism is defined as adding i.i.d. Laplacian noise with variance $\Delta_1 f/\epsilon$ and zero mean to each coordinate value of $f(D)$. Specifically, if $\mathcal{M}(D) = f(D) + [z_1, \ldots, z_n]$, where $z_i \sim \mathcal{L}(\lambda)$ and $\text{pdf}_{\mathcal{L}}(x) = (1/2\lambda)\exp(-|x|/\lambda)$, then mechanism $\mathcal{M}$ provably satisfies $(\epsilon, 0)$-DP [49].

Although the Laplace mechanism suggests that DP can only be used to release aggregate data about a dataset without degrading utility too much, we can apply the same (attack) technique described in Section 7.3 on the noisy aggregates in order to approximate the individual private attribute values and release this approximation as "synthetic" dataset. As this reconstruction step is applied on the already differentially private aggregates as a post-processing step, it does not degrade the privacy guarantee.

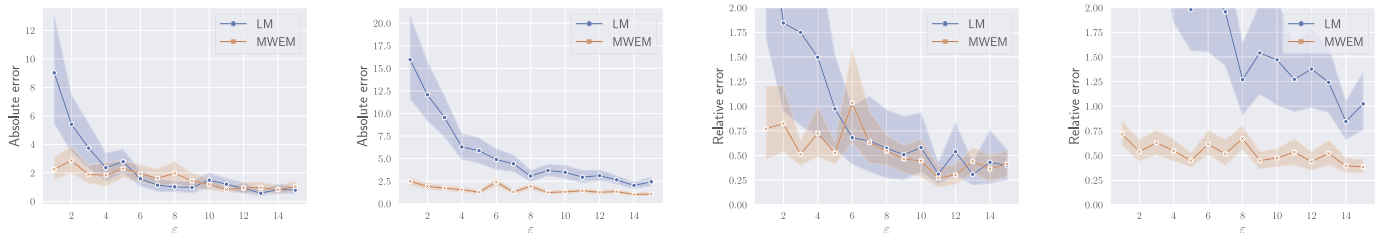
7.4.1. Evaluation

We continue the example from Section 7.3 and attempt to answer counting queries over our dataset while providing the guarantee of Differential Privacy. Counting queries return the number of drivers satisfying a specified predicate over all possible attributes of the dataset. For instance,

```
SELECT COUNT(*) WHERE AGE < 32 AND AGE > 20 AND SEX = 'Male'
AND EXPERIENCE = 1 and AVG_SPEED < 50 AND AVG_SPEED > 30
```

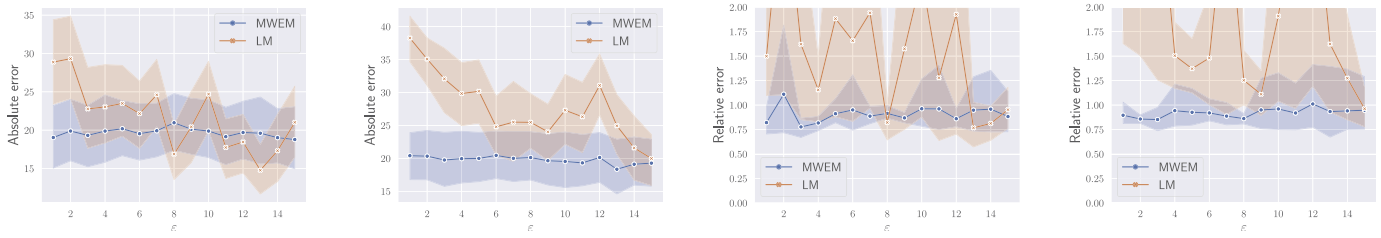
[15] Gauss elimination solves a system of linear equations in $\mathcal{O}(n^3)$ steps, or LU decomposition with Strassen multiplication in $\mathcal{O}(n^{2.8})$ steps. However, in order to unambiguously solve the system, the Q_W matrix has to be non-singular, i.e. each query (row) must be linearly independent. Finding a maximal set of linearly independent query vectors requires $\mathcal{O}(n^2 k)$ time, thus the private $\mathbf{x}$ vector can be reconstructed in polynomial time.

[16] Note that we generate queries only with semantically meaningful predicates, and hence not all queries may be linearly independent. This means that we are likely to generate more than 28 queries for complete reconstruction.



(a) $k = 28$ (b) $k = 112$ (c) $k = 28$ (d) $k = 112$

Fig. 9. Utility metrics for random range queries as a function of ε and the number of queries k.



(a) $k = 28$ (b) $k = 112$ (c) $k = 28$ (d) $k = 112$

Fig. 10. Utility metrics for attribute reconstruction as a function of ε and the number of queries k.

is a valid counting (range) query over all attributes. It easy to see that counting queries are universal, i.e. they can be used to answer any other aggregate queries (such as SUM over any attribute).

Before, we have seen a general recipe to release any query result, that is, adding properly calibrated noise to the answer to be released, also referred to as the Laplace Mechanism (LM): $q'_i(D) := q_i(D) + z_i$, where $z_i \sim \mathcal{L}(\Delta(f)/\epsilon)$, $\Delta(f)$ is the sensitivity of the aggregation function f, and ϵ is the privacy parameter ($\delta = 0$). The sensitivity of any counting query is 1 in *any* dataset, since changing a single driver's data changes any query result with a value at most 1. For k queries that need to be answered independently in an interactive (on-line) manner, this technique guarantees $(k \cdot \epsilon)$-DP due to the composition property of DP. In the non-interactive (off-line) case, when queries are answered in batches and hence are known in advance, a tighter upper bound of the sensitivity of all query answers can be computed by calculating the largest number of queries of the batch that cover the same record.

LM often provides quite inaccurate noisy results if we have too many queries to answer. There are several more sophisticated approaches in the literature [49], here we utilize a standard approach, called MWEM [50]. MWEM only works in the off-line setting, while LM can be applied both in the on-line and off-line cases. MWEM optimizes the answers to all the given queries at once, while LM can answer each query independently and hence consecutively. MWEM creates a synthetic multi-dimensional noisy histogram (data cube) where each bin is assigned to a combination of attribute values and stores the number of occurrences of that combination in the whole dataset.[17] Starting from a rough estimation of this data cube, MWEM iteratively refines that to improve the answers to the given set of queries without violating (ε, δ)-DP. The worst absolute error of LM over all queries scales with $\mathcal{O}(n^{-1}k\log(k))/\varepsilon)$, while MWEM has an error of $\mathcal{O}(n^{2/3}(\log(h)\log(k)/\varepsilon)^{1/3})$ with k queries, n records, and a histogram (data cube) with size h [49,50]. In other words, MWEM overcomes LM if $k >> n$ (i.e., there are significantly more queries than records), or ε is small (i.e., the privacy guarantee is stronger).

We also approximate the private attribute values x_i from the set of noisy aggregates answered by the above two mechanisms. To this end, the following constrained convex optimization problem is solved when LM is used: $\mathbf{x}' = \arg\min_{\mathbf{x}} |\mathbf{Q}_W \cdot \mathbf{x} - \mathbf{q}'|$, where $0 \le x_i \le 60$ for all i, $\mathbf{q}' = [q'_1(D), \ldots, q'_k(D)]$ denote the noisy query results, and $\mathbf{x}'$ are the approximated ("synthetic") private attribute values which correspond to the maximum likelihood estimation (MLE) of x_i when the additive noise is Laplacian. Since MWEM maintains a (noisy) synthetic dataset (or synopsis) that is used to answer a specified set of input queries, the explicit reconstruction of $\mathbf{x}$ is unnecessary and can be retrieved directly from this synopsis.

Utility metrics: Recall that k denotes the number of all queries to be answered. The utility of the released differentially private aggregate and reconstructed "synthetic" data $\mathbf{x}'$ is measured with the following metrics:

- *Average absolute query error*: the average of all absolute distances between the original and the noisy query results; $\sum_{i=1}^{k} |q_i - q'_i|$
- *Average relative query error*: the average of all absolute distances between the original and the noisy query result relative to the original query result; $\sum_{i=1}^{k} \frac{|q_i - q'_i|}{q_i}$
- *Average absolute reconstruction error*: the average of all absolute distances between the original private attribute value and its reconstructed counterpart; $\sum_{i=1}^{n} |x_i - x'_i|$
- *Average relative reconstruction error*: the average of all relative distances between the original private attribute value and its reconstructed counterpart relative to the original value; $\sum_{i=1}^{n} \frac{|x_i - x'_i|}{x_i}$

Results: We generated k random counting range queries for this experiment (i.e., a query predicate includes a random range of every attribute), where k can be 28 or 112, and calculated the four utility metrics as a function of ε. The results are shown in Fig. 9 and 10. The overall tendency is conspicuous, the higher

[17] In our case, the four attributes, i.e. age, sex, experience, average velocity, each discretized into 2, 50, 3, 64 value ranges, respectively, results in a histogram with a size of 19 200.

the ϵ the lower the error gets. Indeed, lower ϵ implies stronger privacy guarantee which in turn requires larger amount of noise. This is especially true for queries with low support typically with small datasets. Nonetheless, a large vehicle management company is likely to work with several magnitude larger number of drivers entailing potentially better utility and making Differential Privacy a more appealing solution. MWEM is clearly superior to LM if the privacy guarantee is stronger (i.e., ε is small) or the number of queries is large. The empirical results are perfectly in line with the theoretical accuracy bounds described above; the difference between the accuracy of MWEM and LM gets larger as the number of queries increases, but smaller if ε increases.

The accuracy of the reconstructed average velocity value per driver is shown in Fig. 10. In general, the reconstruction is inaccurate for both mechanisms due to the small size of the dataset, but MWEM is again more accurate than LM for more stringent privacy guarantees ($\varepsilon < 2$) and has the best relative accuracy of 0.75, whereas LM only falls below 1.0 when the number of queries is relatively smaller. Therefore, even if the accuracy of reconstruction is expected to increase when the number of linearly independent queries also increases, the larger number of queries also have larger sensitivity and therefore require to add a larger amount of noise to the query results in order to satisfy DP.

8. Conclusion

We showed that CAN logs carry personal and sensitive information and are therefore subject to privacy regulations. We first reconstructed the potentially sensitive trajectory of the vehicle from different sensor measurements other than the exact GPS coordinates. Our attack does not rely on any special background knowledge for trajectory reconstruction besides the speed, steering wheel position, and the starting location of the vehicle. In addition, our reconstruction technique is simple and efficient, and can also serve as a useful forensics tool, e.g., in case of accident reconstruction. Our second attack re-identifies the driver among several others with an average accuracy of more than 0.97 utilizing a machine learning model trained on some already available raw CAN data of the driver. As these attacks are feasible under mild assumptions, even unprocessed raw CAN logs are undoubtedly regarded as personal data according to most privacy regulations.

Our attacks exploit the subtle dependencies among different sensor measurements which makes their anonymization, as well as consent control, challenging. Indeed, we also showed that naive de-identification approaches, such as suppression, aggregation, smoothing or low-pass filtering, either do not provide a strong worst-case privacy guarantee to every single individual or they yield very inaccurate data. Therefore, we advocate the application of more principled approaches, such as Differential Privacy, to anonymize aggregated CAN logs, and provide identical privacy guarantees to every single driver irrespective of any background knowledge of the adversary. On the other hand, Differential Privacy generally provides meaningful utility only with a sufficiently large number of drivers, and can be used for synthetic data generation indirectly through the post-processing of already noisy aggregates. Therefore, further research is needed to improve the accuracy of the differentially private release of CAN data with sufficiently strong privacy guarantees.

Finally, instead of data anonymization, data controllers may rather ask for the consent of drivers in order to process or share their vehicular data, and still remain compliant with privacy regulations. Even if fine-grained controls to review and grant permissions to certain signals are provided to drivers with the description of how data will be shared and used, there is usually less explanation on how much data they can share without revealing sensitive information due to the interdependency of different signals or attributes. However, in the spirit of the GDPR, providing *informed* consent should also entail the comprehension of such potential privacy implications.

Our solutions have some limitations. First, our trajectory reconstruction needs to access the exact initial location and heading of the vehicle as well as the steering wheel angle and speed signals extracted from the CAN data. It is an open question how to relax these assumptions without significantly degrading reconstruction accuracy. Second, our driver re-identification technique assumes the availability of some training data about the drivers to be re-identified. Prior works [13] have relaxed this assumption only at the cost of accuracy degradation. Finally, releasing CAN data with Differential Privacy guarantees entails significant utility loss unless the data to be released are adequately large. Further research is needed to provide better privacy-utility trade-off with meaningful privacy guarantees.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

We plan to release our source codes and datasets upon acceptance of our manuscript.

Acknowledgements

Project no. 138903 has been implemented with the support provided by the Ministry of Innovation and Technology from the NRDI Fund, financed under the FK_21 funding scheme. The research was supported by the Ministry of Innovation and Technology NRDI Office within the framework of the Artificial Intelligence National Laboratory Program.

Appendix A. Table of substantial notations

	General
D	Dataset
R	Region
	Microtracking
β	Wheel position (angle)
L	Length of vehicle
R	Radius
	Macrotracking
ω	Weight
γ	Distance threshold to the nearest intersection
C_i	Test case identifier
	Re-identification
K	Number of all time series
T	Time series
s_i	i-th segment of a time series
h_i	i-th hidden state of the LSTM
TP	True positive
TN	True negative
FP	False positive
FN	False negative
P	Number of all positives
N	Number of all negatives
	Defense
x_i	i-th private attribute
$q(D)$	Query on dataset
f	Aggregate function
id_i	i-th identifying information
W	Set of all possible identifiers
(ϵ, δ)	Parameters of differential privacy

Appendix B. Mixture model

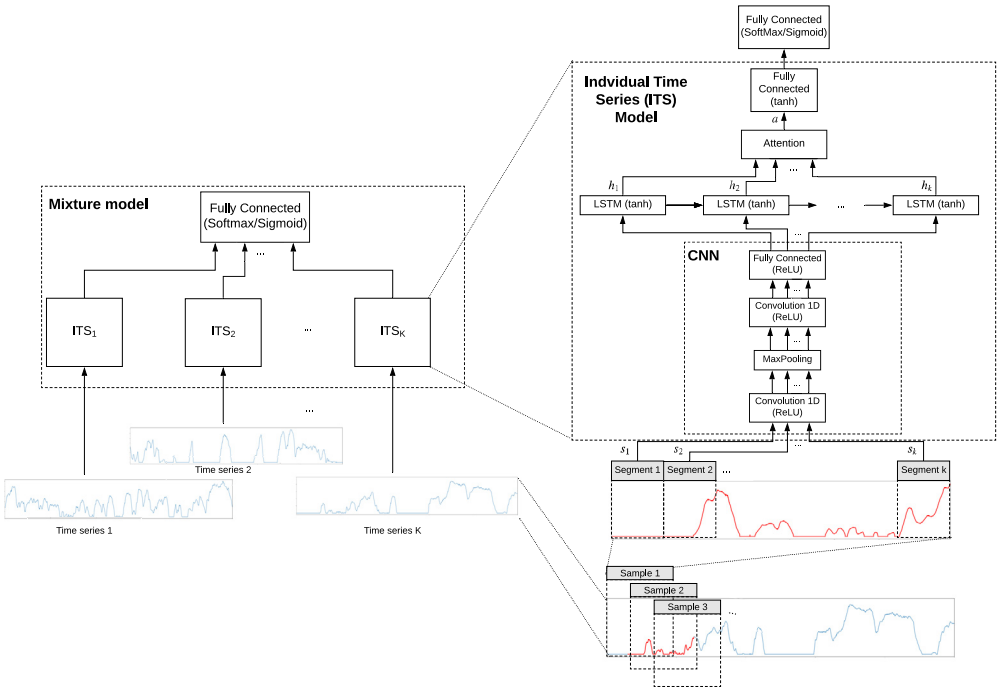
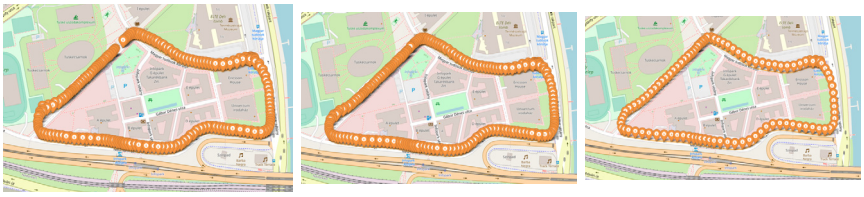


Fig. 11. Mixture model.

Appendix C. Effect of smoothing and low-pass filtering on macrotracking

(a) Smoothing window: 0.201s (b) Smoothing window: 1.608s (c) Smoothing window: 6.4s

Fig. 12. C2 test case macrotracking results on smoothed data without map.



(a) Smoothing window: 0.201s (b) Smoothing window: 1.608s (c) Smoothing window: 6.4s

Fig. 13. C2 test case macrotracking results on smoothed data with map.

(a) Smoothing window: 0.201s (b) Smoothing window: 1.608s (c) Smoothing window: 6.4s

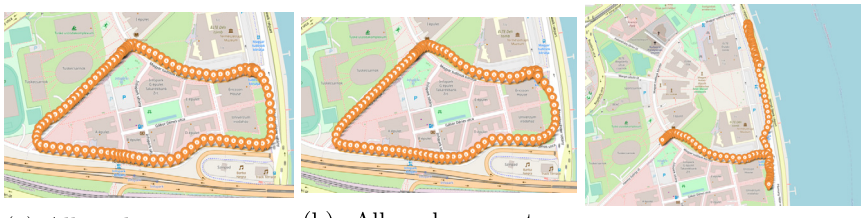
Fig. 14. C3 test case macrotracking results on smoothed data without map.

(a) Smoothing window: 0.201s (b) Smoothing window: 1.608s (c) Smoothing window: 6.4s

Fig. 15. C3 test case macrotracking results on smoothed data with map.

(a) Allowed reconstruction error: 10% (b) Allowed reconstruction error: 20% (c) Allowed reconstruction error: 40%

Fig. 16. C2 test case macrotracking results on low pass filtered data without map.



(a) Allowed reconstruction error: 10% (b) Allowed reconstruction error: 20% (c) Allowed reconstruction error: 40%

Fig. 17. C2 test case macrotracking results on low pass filtered data with map.

(a) Allowed reconstruction error: 10% (b) Allowed reconstruction error: 20% (c) Allowed reconstruction error: 40%

Fig. 18. C3 test case macrotracking results on low pass filtered data without map.

(a) Allowed reconstruction error: 10% (b) Allowed reconstruction error: 20% (c) Allowed reconstruction error: 40%

Fig. 19. C3 test case macrotracking results on low pass filtered data with map.

References

[1] European Parliament and the Council, Regulation (EU) 2016/679 of the European Parliament and of the Council of 27 April 2016 on the protection of natural persons with regard to the processing of personal data and on the free movement of such data, and repealing Directive 95/46/EC (General Data Protection Regulation), Off. J. Eur. Union L119 (May 2016) 1–88.

[2] Aloni Cohen, Kobbi Nissim, Towards formalizing the gdpr’s notion of singling out, Proc. Natl. Acad. Sci. 117 (15) (2020) 8344–8352.

[3] Yves-Alexandre De Montjoye, César A. Hidalgo, Michel Verleysen, Vincent D. Blondel, Unique in the crowd: the privacy bounds of human mobility, Sci. Rep. 3 (1) (2013) 1–5.

[4] Chiyomi Miyajima, Yoshihiro Nishiwaki, Koji Ozawa, Toshihiro Wakita, Katsunobu Itou, Kazuya Takeda, Fumitada Itakura, Driver modeling based on driving behavior and its evaluation in driver identification, Proc. IEEE 95 (2) (2007) 427–437.

[5] Umberto Fugiglando, Emanuele Massaro, Paolo Santi, Sebastiano Milardo, Kacem Abida, Rainer Stahlmann, Florian Netter, Carlo Ratti, Driving behavior analysis through can bus data in an uncontrolled environment, IEEE Trans. Intell. Transp. Syst. 20 (99) (2018).

[6] David Hallac, Abhijit Sharang, Rainer Stahlmann, Andreas Lamprecht, Markus Huber, Martin Roehder, Jure Leskovec, et al., Driver identification using automobile sensor data from a single turn, in: 2016 IEEE 19th International Conference on Intelligent Transportation Systems (ITSC), IEEE, 2016, pp. 953–958.

[7] Kelvin Sopnan Wowo, Michael Nolting, Nicolas Tempelmeier, Towards submaneuver selection for automated driver identification, in: 2022 IEEE 25th International Conference on Intelligent Transportation Systems (ITSC), 2022, pp. 2422–2427.

[8] Umberto Fugiglando, Paolo Santi, Sebastiano Milardo, Kacem Abida, Carlo Ratti, Characterizing the driver dna through can bus data analysis, in: Proceedings of the 2nd ACM International Workshop on Smart, Autonomous, and Connected Vehicular Systems and Services, ACM, 2017, pp. 37–41.

[9] Miro Enev, Alex Takakuwa, Karl Koscher, Tadayoshi Kohno, Automobile driver fingerprinting, Proc. Priv. Enh. Technol. 2016 (1) (2016) 34–50.

[10] Najmeddine Abdennour, Tarek Ouni, Nader Ben Amor, Driver identification using only the can-bus vehicle data through an RCN deep learning approach, Robot. Auton. Syst. 136 (2021) 103707.

[11] Li Zeng, Mohammad Alrifai, Michael Nolting, Wolfgang Nejdl, Triplet loss for effective deployment of deep learning based driver identification models, in: 24th IEEE International Intelligent Transportation Systems Conference, ITSC 2021, Indianapolis, IN, USA, September 19-22, 2021, IEEE, 2021, pp. 1328–1333.

[12] Li Zeng, Mohammad Al-Rifai, Sergiu Chelaru, Michael Nolting, Wolfgang Nejdl, On the importance of contextual information for building reliable automated driver identification systems, in: 2020 IEEE 23rd International Conference on Intelligent Transportation Systems (ITSC), 2020, pp. 1–8.

[13] Jianfeng Li, Kaifa Zhao, Yajuan Tang, Xiapu Luo, Xiaobo Ma, Inaccurate prediction is not always bad: open-world driver recognition via error analysis, in: 2021 IEEE 93rd Vehicular Technology Conference (VTC2021-Spring), 2021, pp. 1–7.

[14] Yijie Xun, Jiajia Liu, Nei Kato, Yongqiang Fang, Yanning Zhang, Automobile driver fingerprinting: a new machine learning based authentication scheme, IEEE Trans. Ind. Inform. 16 (2) (2020) 1417–1426.

[15] Osman Abul, Batuhan Karatas, Can driving patterns predict identity and gender?, J. Ambient Intell. Humaniz. Comput. 12 (1) (2021) 151–166.

[16] Moti Markovitz, Avishai Wool, Field classification, modeling and anomaly detection in unknown can bus networks, Veh. Commun. 9 (2017) 43–52.

[17] Abdul Rehman Javed, Saif ur Rehman, Mohib Ullah Khan, Mamoun Alazab, Thippa Reddy, G. Canintelliids, Detecting in-vehicle intrusion attacks on a controller area network using cnn and attention-based gru, IEEE Trans. Netw. Sci. Eng. 8 (2) (2021) 1456–1466.

[18] Rinku Dewri, Prasad Annadata, Wisam Eltarjaman, Ramakrishna Thurimella, Inferring trip destinations from driving habits data, in: Proceedings of the 12th ACM Workshop on Privacy in the Electronic Society, 11 2013, pp. 267–272.

[19] Xianyi Gao, Bernhard Firner, Shridatt Sugrim, Victor Kaiser-pendergrast, Yulong Yang, Janne Lindqvist, Elastic Pathing: Your Speed Is Enough to Track You, 2014.

[20] Lu Zhou, Qingrong Chen, Zutian Luo, Haojin Zhu, Cailian Chen, Speed-based location tracking in usage-based automotive insurance, in: 2017 IEEE 37th International Conference on Distributed Computing Systems (ICDCS), 2017, pp. 2252–2257.

[21] Lu Zhou, Suguo Du, Haojin Zhu, Cailian Chen, Kaoru Ota, Mianxiong Dong, Location privacy in usage-based automotive insurance: attacks and countermeasures, IEEE Trans. Inf. Forensics Secur. 14 (1) (2019) 196–211.

[22] Vladimir Kaplun, Michael Segal, Breaching the privacy of connected vehicles network, Telecommun. Syst. 70 (04 2019).

[23] Marian Waltereit, Maximilian Uphoff, Torben Weis, Route derivation using distances and turn directions, in: Proceedings of the ACM Workshop on Automotive Cybersecurity, 03 2019, pp. 35–40.

[24] Mert D. Pesé, Xiaoying Pu, Kang G. Shin, Spy: car steering reveals your trip route!, Proc. Priv. Enh. Technol. 2020 (2) (2020) 155–174.

[25] Ankur Sarker, Chenxi Qiu, Haiying Shen, Hua Uehara, Kevin Zheng, Brake data-based location tracking in usage-based automotive insurance programs, in: 2020 19th ACM/IEEE International Conference on Information Processing in Sensor Networks (IPSN), 2020, pp. 229–240.

[26] Jun Han, Emmanuel Owusu, Le T. Nguyen, Adrian Perrig, Joy Zhang, Accomplice: location inference using accelerometers on smartphones, in: 2012 Fourth International Conference on Communication Systems and Networks (COMSNETS 2012), IEEE, 2012, pp. 1–9.

[27] ISO/TC 22 Road vehicles, Road Vehicles – Interchange of Digital Information – Controller Area Network (CAN) for High-Speed Communication, Standard, International Organization for Standardization, Geneva, CH, 1993.

[28] Sara Bressman, Restricting reverse engineering through shrink-wrap licenses: Bowers v. Baystate Technologies, Inc., Boston Univ. J. Sci. Technol. Law 9 (2003) 185.

[29] Mina Remeli, Szilvia Lestyán, Gergely Acs, Gergely Biczók, Automatic driver identification from in-vehicle network logs, in: 2019 IEEE Intelligent Transportation Systems Conference (ITSC), IEEE, 2019, pp. 1150–1157.

[30] Mirco Marchetti, Dario Stabili, Read: reverse engineering of automotive data frames, IEEE Trans. Inf. Forensics Secur. 14 (4) (2018) 1083–1097.

[31] Thomas Huybrechts, Yon Vanommeslaeghe, Dries Blontrock, Gregory Van Barel, Peter Hellinckx, Automatic reverse engineering of can bus data using machine learning techniques, in: International Conference on P2P, Parallel, Grid, Cloud and Internet Computing, Springer, 2017, pp. 751–761.

[32] Miki Elizabeth Verma, Robert Anthony Bridges, Jordan Jeffrey Sosnowski, Samuel C. Hollifield, Michael David Iannacone, Can-d: a modular four-step pipeline for comprehensively decoding controller area network data, IEEE Trans. Veh. Technol. (2021).

[33] Clinton Young, Jordan Svoboda, Joseph Zambreno, Towards reverse engineering controller area network messages using machine learning, in: 2020 IEEE 6th World Forum on Internet of Things (WF-IoT), IEEE, 2020, pp. 1–6.

[34] András Gazdag, Tamás Holczer, Levente Buttyán, Zsolt Szalay, Vehicular can traffic based microtracking for accident reconstruction, in: Vehicle and Automotive Engineering, Springer, 2018, pp. 457–465.

[35] Szilvia Lestyan, Gergely Acs, Gergely Biczók, Zsolt Szalay, Extracting vehicle sensor signals from can logs for driver re-identification, in: 5th International Conference on Information Security and Privacy (ICISSP 2019), SCITEPRESS, 2019, shortlisted for Best Student Paper Award.

[36] T.K. Vintsyuk, Speech discrimination by dynamic programming, Cybernetics 4 (1968) 52–57.

[37] E.J. Lefferts, F.L. Markley, M.D. Shuster, Kalman filtering for spacecraft attitude estimation, J. Guid. Control Dyn. 5 (5) (1982) 417–429.

[38] Greg Welch, Gary Bishop, An introduction to the Kalman filter, Technical Report TR95-041, University of North Carolina at Chapel Hill, Department of Computer Science, 1995. The article has also been translated into Chinese by Xuchen Yao, a student at the Institute of Acoustics of the Chinese Academy of Sciences. See also our Kalman filter web site at https://www.cs.unc.edu/~welch/kalman/index.html.

[39] Geoff Boeing, Osmnx: new methods for acquiring, constructing, analyzing, and visualizing complex street networks, Comput. Environ. Urban Syst. 65 (2017) 126–139.
[40] Brent C. Nolan, Scott Graham, Barry Mullins, Christine Schubert Kabban, Unsupervised time series extraction from controller area network payloads, in: 2018 IEEE 88th Vehicular Technology Conference (VTC-Fall), IEEE, 2018, pp. 1–5.
[41] George Ellis, Filters in control systems, Control Syst. Des. Guide 9 (2012) 165.
[42] Alice Cohen-Hadria, Mark Cartwright, Brian McFee, Juan Pablo Bello, Voice anonymization in urban sound recordings, in: 2019 IEEE 29th International Workshop on Machine Learning for Signal Processing (MLSP), IEEE, 2019, pp. 1–6.
[43] Shubha U. Nabar, Krishnaram Kenthapadi, Nina Mishra, Rajeev Motwani, A survey of query auditing techniques for data privacy, in: Privacy-Preserving Data Mining, Springer, 2008, pp. 415–431.
[44] Simson L. Garfinkel, John M. Abowd, Christian Martindale, Understanding database reconstruction attacks on public data, Commun. ACM 62 (3) (2019) 46–53.
[45] Francis Y. Chin, Gultekin Ozsoyoglu, Auditing and inference control in statistical databases, IEEE Trans. Softw. Eng. SE-8 (6) (1982) 574–582.
[46] Irit Dinur, Kobbi Nissim, Revealing information while preserving privacy, in: Proceedings of the Twenty-Second ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems, 2003, pp. 202–210.
[47] Cynthia Dwork, Frank McSherry, Kobbi Nissim, Adam Smith, Calibrating noise to sensitivity in private data analysis, in: Theory of Cryptography Conference, Springer, 2006, pp. 265–284.
[48] Cynthia Dwork, Krishnaram Kenthapadi, Frank McSherry, Ilya Mironov, Moni Naor, Our data, ourselves: privacy via distributed noise generation, in: Annual International Conference on the Theory and Applications of Cryptographic Techniques, Springer, 2006, pp. 486–503.
[49] Cynthia Dwork, Aaron Roth, et al., The algorithmic foundations of differential privacy, Found. Trends Theor. Comput. Sci. 9 (3–4) (2014) 211–407.
[50] Moritz Hardt, Katrina Ligett, Frank McSherry, A simple and practical algorithm for differentially private data release, in: Peter L. Bartlett, Fernando C.N. Pereira, Christopher J.C. Burges, Léon Bottou, Kilian Q. Weinberger (Eds.), Advances in Neural Information Processing Systems 25: 26th Annual Conference on Neural Information Processing Systems 2012, Proceedings of a Meeting Held December 3-6, 2012, Lake Tahoe, Nevada, United States, 2012, pp. 2348–2356.