Using Phone Sensors to Augment Vehicle Reliability

Noah T. Curran*
University of Michigan
ntcurran@umich.edu

Arun Ganesan*
Meta
arunganesan@meta.com

Mert D. Pesé
Clemson University
mpese@clemson.edu

Kang G. Shin
University of Michigan
kgshin@umich.edu

***Abstract*— The increasing connectivity of vehicles has led to wide exploitation of their vulnerability/unreliability surface. As a result, the security and reliability of vehicle sensor information has become a pressing concern because of the importance of sensor information to vehicular functions. To address this concern, state-of-the-art anomaly detectors validate vehicle sensors via the information *internal* to a vehicle. However, they are still prone to data unreliability as vehicles are built with little to no sensor redundancy.**

To further enhance the reliability of vehicle sensors, we present `CaRe`,[1] which uses the driver's smartphone as an *external* source of sensor redundancy for detecting vehicle sensor anomalies. `CaRe` uses sensing capabilities available in smartphones to estimate vehicle sensor values and ensures smartphone sensing to be resilient to common phone usage to improve the accuracy of estimations. It logs anomalies and informs the driver of detected anomalies to limp home and/or to inspect later.

Using injections on vehicle sensor traces, `CaRe` is empirically shown to detect anomalies from 5 safety-critical vehicle sensors with high true positive rates (for sudden injections, range from 97.33% for speed to 90.08% for gear) while keeping the false positive rates very low (almost always less than 1%). `CaRe` can estimate vehicle sensors and detect anomalies with low computational overhead (≈8% CPU usage) and low time latency (bottlenecked by the sensor refresh rate).

I. Introduction

In the last decade, the increased connectivity of vehicles has enabled a wide range of vehicular functions/applications that allow for a more seamless driving experience, but it accompanied unexpected/undesired consequences. Exposing the vehicle to a vast arrangement of wireless communication technologies has widened the surface for malicious actors to infiltrate the vehicle and inject incorrect data into its sensors [1]–[6].

While physical sensor failures pre-date vehicle hacks, the remote vehicle hack in [6] motivated the rapid expansion of research that detects and defends against anomalous communications within the *Controller Area Network* (CAN), the *de facto* standard of in-vehicle networks. Thus, research on vehicular *Intrusion Detection System* (IDS) has dominated the landscape and steered in several different directions, including voltage fingerprinting [7]–[13], packet modeling [14]–[17], sensor correlation [18]–[20], and machine learning [21], [22], as described in § II-B. While these efforts have advanced the state of vehicular security and reliability, vehicles still remain vulnerable since their internal networks are built with little to no redundancy. Therefore, should an adversary gain write-access to the CAN of a vehicle, they would be able to directly intrude or deceive the computer that hosts IDSs since there is no redundancy for the IDSs to validate against. Furthermore, there is no indication this practice will change anytime soon due to stringent financial constraints that automotive OEMs impose on vehicle manufacturing.

Left with the realistic constraint that the automotive industry will not equip vehicles with sensor redundancy, we have limited options *inside* the vehicle network. In order to provide redundancy in automotive networks at no extra cost, one can naturally seek redundancy through compute and sense resources that are pervasively available *outside* the vehicle network. Following this anticipation, we present `CaRe`, a mobile app that utilizes smartphone resources (*e.g.*, accelerometer, gyroscope) to sense vehicular dynamics and accurately estimate vehicular sensor values. To use these estimations as a substitute for sensor redundancy, `CaRe` communicates with the vehicle's *On-Board Diagnostics* (OBD-II) port to cross-validate the estimations against the data values on the CAN. (We discuss OBD-II security concerns in § III-B.) When the difference between the two values exceeds a trained threshold for a trained duration of time, the detected anomaly is logged for inspection. While outside the scope of this paper, we briefly discuss what to do with detected anomalies in § VIII. In pursuit of providing sensor redundancy at no cost through `CaRe`, we make the following main contributions:

1) Introduction of a *new* and *practical* layer of redundancy for vehicles through ubiquitous external technologies;
2) Provide smartphone sensing resilient to common phone usage in order to measure vehicle dynamics and *estimate the status of vehicular sensors* (§ IV and § V);
3) Design a framework that combines vehicle and smartphone sensors to *detect anomalies* and *enhance vehicle sensor reliability* (§ VI);
4) Propose a deployment scenario for and application of `CaRe` that reinforces its practicality (§ VIII).

To the best of our knowledge, `CaRe` is the first attempt at augmenting vehicle reliability through indirect redundancy gained from ubiquitously available *external* resources without incurring additional vehicle manufacturing costs. `CaRe` has virtually no added costs since those concerned with vehicle diagnostics likely already own an OBD-II dongle to interface with the CAN and since smartphone ownership is ubiquitous in today's society. Furthermore, `CaRe` is the first vehicle-oriented smartphone app that diagnoses vehicle sensor anomalies through smartphone sensing capabilities. Both end-users (*i.e.*, drivers and passengers) and car mechanics would use `CaRe` diagnose vehicular sensor anomalies.

Moreover, compared to other vehicular IDSs, `CaRe` requires no reference model to detect anomalous sensor data since it only utilizes smartphone sensors to estimate the true vehicle sensor values. Finally, while `CaRe` is shown to incur an acceptable level of power consumption (≈8% of the CPU on average), the smartphone will remain in-vehicle and can thus utilize the car battery for charging while driving.

*These authors contributed equally to this work. Arun made his contributions while at the University of Michigan

[1]Pronounced like "care," `CaRe` stands for "Car Reliability."

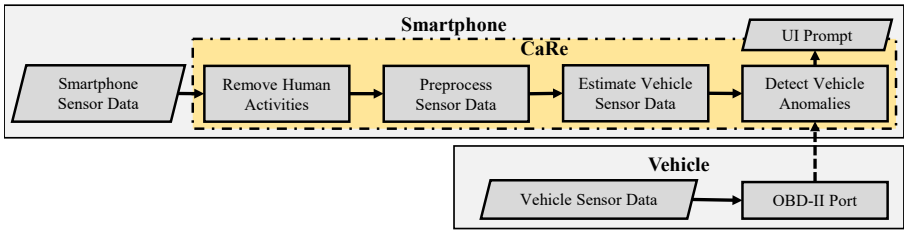


Fig. 1: An overview of the system design of CaRe, a smartphone app for augmenting vehicle reliability.

We implement CaRe in Android phones and evaluate it against several classes of anomalies. To best evaluate CaRe, we create a taxonomy of anomalies reported in the literature—*sudden*, *gradual*, and *delta* (§ III-A). Our evaluation of CaRe injects these three types of anomalies into real-world driving traces and shows CaRe to be capable of detecting anomalies of 5 vehicular sensors with different levels of true positive rate (TPR), ranging from 97.33% for speed values to 90.08% for RPM values, with very low false positive rate (FPR), often less than 1% FPR. CaRe can accurately detect speed, gear position, fuel and odometer anomalies, with the detection latency bottlenecked by the sensor refresh rate.

II. Background & Related Work

A. Vehicle Dynamics Estimation

Recent work has shown how to use a vehicle's dynamics to estimate its state, and then subsequently use the estimation to determine whether the vehicle is behaving abnormally [23]. While orthogonal to our work, we neither directly estimate vehicle dynamics nor track the vehicle state. Rather, we inspect individual estimations of vehicle sensor values.

B. Vehicular Intrusion Detection

Prior work on vehicle sensor reliability focused primarily on the cybersecurity problem of detecting adversarial intrusion [24]. In contrast, we discuss the different lines of work in this direction, highlighting CaRe's novelty of incorporating an *external* source of redundancy for detecting anomalies.

1) Voltage Fingerprinting: One class of IDSs utilize voltage fingerprinting [7]–[13]. For instance, in [8] the IDS used clock-based fingerprinting of ECUs on the CAN to identify any that are misbehaving. This work is orthogonal to CaRe, as it can act as a layer of security for locating the intrusion once it is detected by CaRe.

2) Packet Modeling: There have also been IDSs that model the information-theoretic and structural patterns of the CAN under normal behavior [14]–[17]. During an attack, they observed that information-theoretic properties, such as entropy, are likely to change. However, they fail to detect attacks that do not change CAN traffic [25], [26]. The information-theoretic properties change only for CAN injection attacks that deviate from normal behavior of the CAN. If the adversary is able to mount the attack without changing the CAN's behavior—such as through a bus-off attack [25] or Bootrom attack [26]—then it may evade detection. In contrast, CaRe does not model the CAN traffic.

3) Sensor Correlation: Other IDSs model the normal behavior of the vehicle by comparing with other vehicle sensors' data on the CAN [18]–[20], [27]. These approaches rely on internal vehicle sensors that may be susceptible to the same adversary who can inject data into the CAN. While this category of vehicular IDSs is most similar to that of CaRe, CaRe uses an *external* source of redundancy for validating the sensors within the vehicle.

4) Machine Learning: Similar to packet modeling, the final set of solutions utilize machine learning to characterize normal behaviors of the CAN [21], [22]. These solutions are often computationally expensive and are not as effective on the computationally limited and outdated ECUs that are embdedded within vehicles. Furthermore, CaRe provides an external source of validation and does not provide a characterization for what is or is not normal traffic for the CAN bus, and is not vulnerable to attack classes relevant to machine learning solutions.

III. Threat Model

A sensor anomaly occurs when a sensor reports a value that does not match what it is meant to observe. This may happen when sensors become faulty, physically damaged, or compromised by an adversary.

We first describe how an adversary may launch a sensor-falsification attack on the CAN. An adversary must first gain write-access to the CAN since the CAN is a broadcast network [3], [26]. Regardless of whether a targeted sensor is a component of the compromised unit, information for the targeted sensor can be spoofed, falsified, and broadcast on the CAN from the compromised entry-point. There have been a variety of methods for obtaining write-access to the CAN [2]–[4], [6], [26], [28]–[30]. While these attack vectors have been exploited widely, other hypothetical scenarios are also considered, such as physical sensor attacks.

One goal of sensor-falsification attacks is to take control of a vehicle, which allows an adversary to perform a number of actions, such as draining the vehicle's battery [31], controlling the steering direction [5], or performing another attack listed in Tbl. I. Such control of the vehicle may cause physical harm to either the vehicle or the cargo and passengers inside.

A. Taxonomy of Anomalies

Despite extensive research on vehicle reliability and the often-used *sensor injection* for compromising a vehicle in the attack literature (see Tbl. I), there is a lack of standard evaluation metrics for detecting vehicle anomalies. We survey existing literature on vehicular cyberattacks and create a taxonomy of sensor anomalies to create this standard. Due to the similar nature of sensor attacks and physical sensor failures, this taxonomy is applicable to either scenario.

Later in this paper we will utilize this taxonomy to evaluate the effectiveness of CaRe. Our taxonomy uses the notation x_t and x'_t for the t^{th} real sensor value and injected sensor value.

1) Sudden Anomaly: In a *sudden anomaly* the CAN is flooded with a single value $x'_0 = ... = x'_n$, squashing the real values following x_0. Adversaries have used this to flood the CAN with injected engine RPM values using an attack to put the engine ECU into diagnostic mode, preventing it from communicating with other ECUs on the CAN [26].

Ref.	ID	Target ECU	Sensor spoofed	Purpose of attack
[4]	1	Instrument Cluster (IC)	Speed	Confuse user
	2	IC	Odometer	Confuse user
	3	Intel. Park. Assistant System (IPAS)	Gear, Speed	Control steering
	4	IC	Fuel gauge	Confuse user
[3]	5	IC	Fuel gauge	Confuse user
	6	IC	Speed	Confuse user
[26]	7	IC	Speed	Confuse user
	8	Park. Assistance Module (PAM)	Speed	Control steering

TABLE I: Example CAN injection attacks which require falsifying vehicular sensors.

2) Gradual Anomaly: An anomaly may steadily and stealthily change the real value to a target value over a sequence of gradual changes in what we call a *gradual anomaly*. For a series of sensor values, a value c_i is added to the real sensor value. The sequence of values in the anomaly hence becomes $x'_0 = x_0 + c_0, ..., x'_n = x_0 + c_n$. In the literature, this anomaly is favored in attack scenarios where stealth is required, such as to cause driver and/or passenger confusion or to be within physical limits of a vehicle safety feature [3], [4].

3) Delta Anomaly: In a *delta anomaly* the falsified values are consistently offset from the real values following x_0 by a value c. Therefore, real values $x_0, \ldots, x_n$ are altered to be $x'_0 = x_0 + c, ..., x'_n = x_n + c$. An example of this is in [3], where the adversary falsifies the speedometer to report a value exactly 10 mph below the real speed of the vehicle.

B. Smartphone Connectivity Security

To interface with the CAN from a smartphone, we utilize the OBD-II port beneath the steering wheel through a dongle. While some OBD-II dongles have previously been shown to contain wireless vulnerabilities for obtaining write-access on the CAN [32], we assume the OBD-II dongle used by `CaRe` is one that is unsophisticated and inexpensive, lacking direct Internet connectivity and programmability available in vulnerable OBD-II dongles. OBD-II dongles of this variety pair with the smartphone through Bluetooth or BLE—which can only pair with one device at a time (*i.e.*, the smartphone that hosts `CaRe`)—and can only listen to the CAN to relay the sensor data back to the smartphone. Because unsophisticated and inexpensive OBD-II dongles are more accessible to the average consumer, we deem this attractive and sufficient.

Finally, because there exist a plethora of smartphone security solutions [33], [34] and smartphone security is not the goal of augmenting vehicular redundancy, we assume the smartphone that runs `CaRe` to follow best practices for security and for `CaRe` to remain uncompromised. However, should a clever adversary find a way to compromise `CaRe`, the adversary would still need to find a way to compromise the CAN communication alongside compromising the smartphone due to the fact that the smartphone lends itself as a source of redundancy.

IV. Removing Phone Sensor Noise

Before using smartphone sensing information for vehicle sensor value estimations, we first pre-process the collected sensor values. Namely, we ensure the smartphone sensing is resilient to driver/passenger phone usage unrelated to vehicular dynamics.

In contrast to prior work that estimates vehicular sensors using phone sensors [35]–[40], we explore if phone sensing can be used to enhance vehicle reliability.

A. Vehicular Mobile Sensing

To detect vehicular dynamics from a smartphone, we make use of the set of sensors in the *Inertial Measurement Unit* (IMU), consisting of the accelerometer, gyroscope, and magnetometer. To use the accelerometer and gyroscope for `CaRe`'s estimations, we first align the phone's readings to the vehicle's direction of travel. There are two methods for phone-to-vehicle alignment that we consider: one that makes use of the GPS and one that does not make use of the GPS. The rationale for this is that there are some cases where the GPS is incapable of providing an accurate alignment due to noise from urban canyons (Fig. 4b).

For the first method, given consecutive GPS points, we find the angle of the GPS bearing offset from the magnetic north. We then rotate the magnetic north vector from the magnetometer by the same angle to get the vehicle pointing vector from the phone's frame of reference, called $\vec{V}$. We do this using a change of basis transformation from the plane perpendicular to the frame of reference which has basis vectors $\vec{M}, \vec{G}, \vec{G} \times \vec{M}$, where $\vec{M}$ is the magnetic north and $\vec{G}$ is the direction of gravity. Using $\vec{G}$ and $\vec{V}$ we can calculate the rotation matrix $\boldsymbol{R}$ by the following equation:

$$\vec{C} = \vec{V} \times \vec{G}; \quad \boldsymbol{R} = \begin{bmatrix} \vec{C_0} & \vec{C_1} & \vec{C_2} \\ \vec{V_x} & \vec{V_y} & \vec{V_z} \\ \vec{G_x} & \vec{G_y} & \vec{G_z} \end{bmatrix} \tag{1}$$

For the second method, we make use of only the accelerometer readings to orient the other IMU readings [41], [42]. In this method, we utilize Euler angles to perform a pre-rotation (ϕ) about the Z-axis of the vehicle, a tilt (θ) about the X-axis of the vehicle, and finally a post-rotation (α) about the Z-axis once again. We obtain the angles for the first two rotations from the stationary vehicle to determine the direction of gravity. For derivations we refer the reader to [41]. Note that in our derivations, $c_x = \cos(x)$ and $s_x = \sin(x)$. Using the accelerometer reading (a_x, a_y, a_z), the equations are:

$$\phi = \arctan(a_x / a_y), \quad \theta = \arccos(a_z). \tag{2}$$

Once the vehicle begins motion, we calculate the third rotation into the direction of acceleration of the vehicle. Using the accelerometer reading (a'_x, a'_y, a'_z), the equation is:

$$\alpha = \arctan\left(\frac{-a'_x c_\phi - a'_y s_\phi}{c_\theta(a'_y c_\phi - a'_x s_\phi) + a'_z s_\theta}\right). \tag{3}$$

We then get rotation matrix $\boldsymbol{R}$ through Euler angle rotations:

$$\boldsymbol{R} = \boldsymbol{R}_Z(\alpha) \times \boldsymbol{R}_X(\theta) \times \boldsymbol{R}_Z(\phi). \tag{4}$$

B. Determining Phone Use

Before estimating vehicle sensor information, we must determine whether the smartphone is in use by a driver or passenger of the vehicle. If it is in use, then the noise caused by user-to-phone interactions will obscure the true nature of the vehicle dynamics, preventing accurate estimations. Although

Sensor	Estimation Method
IMU-align	$\boldsymbol{R} = [(\vec{V} \times \vec{G})^\top; \vec{V}^\top; \vec{G}^\top]$
Speed	$v_t = \alpha(v_{t-1} + Acc_{v,Y} * dt) + (1-\alpha) GPS_v$ $v_t = v_{t-1} + Acc_{v,Y} * dt$
Gear	Neural-network based on vehicle speed
SWA [43]	$\theta_{SWA} = k * arcsin(l * yaw_{rate}/v)$
Odometer	Haversine sum of consecutive GPS
Fuel level	Distance * Average MPG

(a) Estimation of 5 different sensors used in CaRe. For details on each sensor, see their respective sections below.

Speed Offset in Seconds
{-0.5, -0.4, -0.3, -0.2, -0.1, 0}
{-5, -0.5, -0.4, -0.3, -0.2, -0.1, 0}
{-1, 0}
{-3, -2, -1, 0}
{-4, -3, -2, -1, 0}

(b) Each feature vector represents the vehicle's speed sampled at different time offsets in seconds. For each vehicle, we searched through each of these to find the most optimal one.

TABLE II: Summary of estimation equations.

a survey of drivers shows that just 3% of drivers keep their smartphone in-hand while driving [44], this is significant enough to warrant careful handling. Furthermore, while prior work determines whether a driver has a smartphone in-hand for the purpose of driver safety [45]–[47], to the best of our knowledge there is no prior activity recognition work on determining whether any person in a vehicle is holding a smartphone.

Prior work on activity recognition shows that a classifier is sufficient for detecting smartphone use [48]–[51]. Features extracted from IMU sensor readings provide suitable values for classifiers due to their ability to measure environmental motions, and all three IMU sensors can be useful in different situations [52]. Before extracting the features, we need to slice the data into evenly spaced segments. [48], [49] demonstrate that 5-second slices are sufficient for activity recognition. Therefore, utilizing the 12 features listed in [53], we experimentally determine which IMU sensors and classifier combination would be most suitable for classifying if a person is interacting with a smartphone while in a vehicle.

To create the classifier, we gather smartphone IMU sensor data from multiple locations in a vehicle while the smartphone is out-of-hand—such as on a seat, in a bag, in a cup holder, or mounted—and is in-hand. The locations chosen for when the smartphone is out-of-hand are based on popular locations drivers keep their phone while driving [44]. Then, we use the collected data to train five different classifiers with the various IMU sensor combinations of accelerometer, gyroscope, and magnetometer readings. The classifiers chosen are Naïve-Bayes, Logistic Regression, k-Nearest-Neighbor, Rule-Based, and Decision Tree. We evaluate the classifiers in § VII.

C. Phone Sensor Filtering

Once the smartphone is positioned at a suitable location, we proceed with processing the IMU sensor readings. We start with filtering the sensor values to remove noise attributed to the vehicle itself. Li *et al*. [42] find accelerometer readings only need a 2Hz low-pass filter to remove noise that is not attributed to normal vehicle acceleration. The noise removed is from intrinsic high frequency vibrations sourced from thermal and mechanical components of the sensor, and contextual noise from vehicle vibrations.

Error present in the gyroscope can be attributed to a few sources each of which must be addressed individually. First, the gyroscope experiences natural drift over time, causing the values to be off-calibration. This occurs due to DC bias and angular random walk. Second, gyroscope readings have noise from environmental vibrations. Finally, there is noise from the true values attributed to the actual vehicular dynamics, such as turns and lane changes.

To correct the drift, the gyroscope must periodically be re-calibrated. This can be accomplished when the vehicle is determined to be stationary. To distinguish whether reported values from the gyroscope are attributed to the vehicular dynamics or environmental noise, we can use *discrete wavelet transform* (DWT) functions [42], [54], [55]. Finally, when the smartphone shifts due to lane changes or turns, the frame of reference calculated may be no longer accurate. To remedy this, we track the gyroscope filtered through a DWT to determine the angle that the frame of reference changes. We then apply this angle change to the the frame of reference originally calculated [42].

V. Estimating Vehicular Sensors

We estimated 5 in-vehicle sensors using smartphone sensors. The chosen sensors are falsified in various attacks reported in the literature (*cf.* Tbl. I). Furthermore, they are related to vehicle dynamics, thereby making it possible to replicate and cross-validate them using the smartphone sensors. Their estimation equations are summarized in Tbl. IIa.

For some of the estimations, the equations include other estimated vehicle sensors as input. The vehicle sensor estimations may depend upon another estimation, so we must pay close attention to these dependencies, as error accumulated within an estimation can propagate to a dependent estimation. After estimation, CaRe cross-validates the estimations with the CAN-reported sensor values to detect and report anomalies. We give the details of the 5 sensor estimation techniques below.

Some of these sensor estimations require *per-vehicle* calibration. For example, we trained a different neural network for each vehicle in our dataset for Gear estimation, and loaded vehicle-specific parameters for fuel MPG. These must be calibrated one time for each vehicle model and can be performed by the OEM before release. CaRe does not require *per-smartphone* or *per-vehicle* calibration. The same calibration can be loaded on different vehicles of the same model and different smartphones.

Speed. We either fuse the speed estimates from both the accelerometer and GPS sensors using a complementary filter, or use only the accelerometer speed estimates when GPS readings are too noisy. The integration of consecutive accelerometer readings is an estimate of the speed, but due to the noise in the IMU sensor readings, this can result in divergent and incorrect speed estimates. When possible, we use the GPS sensor for speed estimates in the order of 1Hz, but it misses more frequent changes which might be sensed by the IMU sampling at 10Hz.

To use the accelerometer, we first align the phone's IMU accelerometer readings to the vehicle's direction of travel. See § IV-A for an in-depth discussion of this.

As in [43], we also found that the GPS-estimated speed is slightly delayed from the actual vehicle speed. We aligned the GPS-speed by shifting it by ≈0.5 second, a value found experimentally in our data. We fuse the Y axis of the aligned Acc_v, and the delay-adjusted GPS_v using a complementary

filter: $v_t = \alpha(v_{t-1} + Acc_{v,Y} * dt) + (1-\alpha)GPS_v$. We set dt to 100 ms since our accelerometer sample rate for CaRe is 10Hz. We search through a training set and set α to 0.33. Alternatively, we can use just the accelerometer component when the GPS readings are too noisy: $v_t = v_{t-1} + Acc_{v,Y} * dt$.

Gear Position. We focus on automatic transmission vehicles and exclude continuous variable transmission systems. The gear position in automatic transmission vehicles is controlled by the Transmission Control Unit (TCU). The TCU uses inputs from a variety of vehicle sensors to inform its algorithm to upshift or downshift the gear. These sensors include the vehicle speed, throttle position, and many others. The TCU adjusts the gear position to reduce load on the engine, increase safety of the driver, and reduce the long-term wear and tear of internal components.

Since the smartphone lacks many of these sensor values, we trained neural networks using the vehicle's recent change in speed as the feature vector to predict the current gear position. For each vehicle, we found the most accurate feature vector from those listed in Tbl. IIb. We searched through a neural network of depths 1, 2 or 3 where each layer is densely connected with 10 neurons. The output is a one-hot encoding of the current gear position. We used Tensorflow to train these models, and Tensorflow Lite to run them on Android [56].

Steering Wheel Angle. We estimate the steering wheel angle (SWA) using the yaw-rate of the gyroscope on the phone. To accurately estimate the SWA, we align the phone's coordinate system to the world coordinate system (see § IV-A), and then convert the angular rotation in the yaw axis to SWA. We use a similar rotation matrix as $\boldsymbol{R}$ described in § IV-A. Since we are only concerned about the yaw-rate, this only uses the third row of the rotation vector—the vector pointing in the direction of gravity. We call this rotation matrix that only makes use of the gravity vector $\boldsymbol{R}'$. Only using the gravity vector for world-frame alignment is more robust than using the vehicle-facing vector and is sufficient for SWA calculations.

With this new rotation matrix $\boldsymbol{R}'$, we calculate the aligned gyroscopic movement in the world frame of reference by $\vec{g_w} = \boldsymbol{R}' \vec{g_p}^{\top}$ where $\vec{g_w}$ and $\vec{g_p}$ are the gyroscope vectors in the world and phone frame of reference, respectively.

Once we aligned to the world frame of reference, we use a simplified Ackerman mechanism model to estimate the SWA using the smartphone's IMU sensors [43]. The SWA is estimated using $\theta_{steering} = k * arcsin(l * yaw_{rate}/v)$, where k is the steering ratio, l is the vehicle length,[2] v is the vehicle speed and yaw_{rate} is the rotated yaw-rate.

Odometer. For the odometer, consecutive GPS readings are taken after the trip begins, and distances between the GPS readings are summed together. We calculate these distances using the Haversine of consecutive GPS points [57]. From this, we obtain the distance and can add it to the initial odometer reading.

At the start of the trip, we trust the initial odometer value from the CAN to determine how much it has increased. Even if the starting odometer value is anomalous, it would not disrupt future estimations to determine if further anomalies occur.

Fuel Level. We estimate the vehicle's fuel level by using the manufacturer's published average MPG. Starting with a full tank of gas, CaRe uses the manufacturer-published datasheet on the tank capacity of the vehicle. As the user drives his/her vehicle, CaRe matches each location of the vehicle to a road segment[3] and labels that as either highway or city-level driving, using publicly available information [59]. We take the average MPG for the identified type of road and multiply it by the distance traveled to obtain the new fuel tank level.

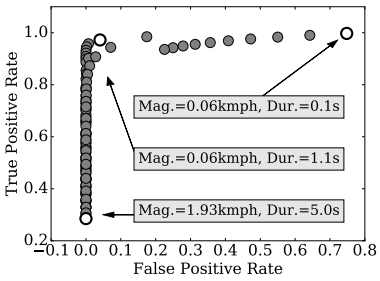


Fig. 2: ROC curve for the speed sensor of 100 combinations of the two parameters—magnitude and duration. For each combination, we calculate the FPR and TPR. An ROC curve was computed for the 5 sensor estimations.

VI. Detecting Vehicular Anomalies

Once we have estimated the vehicular dynamics, we compare them against the reported values on the CAN. Because we cannot perfectly account for and remove all error and noise introduced within the IMU sensors, we must tolerate some margin of error when detecting anomalies. Therefore, CaRe uses two parameters to determine if it should flag an anomaly on the CAN—the magnitude and duration.

The magnitude and duration should both be optimized to unique values for each vehicular sensor estimation. The constraints for the optimization are high *True Positive Rates* (TPRs) and low *False Positive Rates* (FPRs). Given the portion of data that is actually anomalous, the TPR is the fraction of anomalies that are detected (higher is better). For the portion of the data that is normal, the FPR is the fraction of normal data that is incorrectly flagged as anomalous (lower is better).

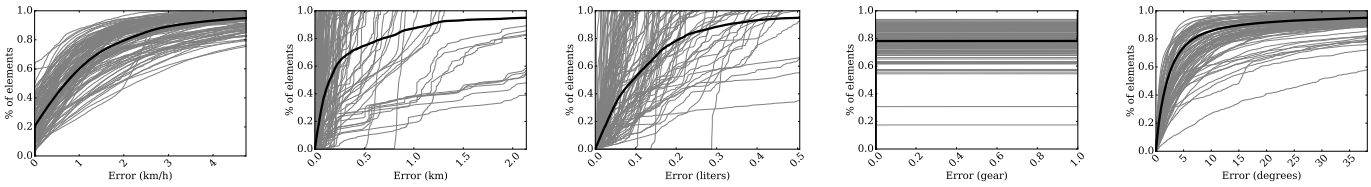
Anomaly Magnitude. The anomaly magnitude is used to determine the difference between the estimated sensor reading, S_{est}, and the CAN-reported reading, S_{rep}. The value from the CAN is flagged as potentially anomalous when the optimized threshold, $D_{threshold}$, is exceeded: $D_{threshold} < |S_{est} - S_{rep}|$. This could potentially be caused by either an attack, a faulty vehicle sensor, or an estimation inaccuracy from the smartphone. Because of the possibility of an estimation inaccuracy, we require a second parameter.

Anomaly Duration. We use a second metric to see how long such a deviation is sustained, measured in seconds. We train a time threshold for flagging a sustained difference as an anomaly.

For each individual vehicular sensor, we search through 100 combinations of both parameters and, in the absence of anomalies, calculate the *receiver operating characteristic* (ROC) curve. Making use of the vehicular sensor estimations evaluated in § VII-B, we set the *magnitude* threshold to one of 10 different values, defined independently for each sensor, and set the *duration* threshold to one of 10 different values equally ranging from 100ms to 5s. For example, Fig. 2 shows the ROC curve for the

[2]We found these in vehicle specifications published online by the manufacturers.

[3]We matched it to OpenStreetMap [58].



(a) Speed: {0.69, 4.74}kmph (b) Odometer: {0.15,2.14}km (c) Fuel: {0.09, 0.51} gallon (d) Gear: {0, 1} gear (e) SWA: {2.02, 38.2}°

Fig. 3: CDF of CaRe estimation error of vehicular sensors during each trip. The average error is the bold line. Captions include the {50th, 90th} percentile error.

Vehicle model	Trips	Hours
Mid-size sedan 2018	21	7.08
SUV A 2017	12	4.87
Compact sedan A 2017	2	0.91
SUV B 2016	44	9.73
Hatchback 2016	2	0.73
Compact sedan B 2012	31	5.07
Total	112	28.56

TABLE III: Driving dataset collected for evaluation of CaRe. We use OpenXC [60] to access the CAN data in all test vehicles. We collected a total of 712.8 miles of data.

speed sensor. With the ROC curve, we search for the configuration which yields the maximum TPR for bounded FPR values, where FPR is bounded to $\{0.1,0.5,0.01,0.001\}$. We discuss the results utilizing the optimized combination of parameters in § VII-C.

VII. Evaluation

A. Accuracy of Phone-Usage Detection

We train and individually evaluate five classifiers in their ability to recognize phone-usage while in-vehicle: Naïve-Bayes, Logistic Regression, k-Nearest-Neighbor, Rule-Based, and Decision Tree. To train these classifiers, we collect smartphone IMU sensor data for various activities in the vehicle. While extracting features from the sensors, we slice data in sizes of 0.5 sec., 1.0 sec., and 5.0 sec. Across the difference combinations of the three sensors in the IMU and the three difference data slice sizes, we train a total of 105 different models. To efficiently train these models, we use a machine learning and data mining software suite [61].

The IMU data is collected in a 2020 Jeep Compass across a variety of road types, including neighborhoods, suburban streets, backroads, and highways. We collected ≈15 minutes of IMU sensor data for seven locations in the vehicle, which were chosen based on typical places smartphone owners keep their phone while driving [44]. The seven locations we place the phone during data collection are: mounted, cup-holder, seat, bag, shirt pocket, pant pocket, or held. The IMU sensor data is collected at a rate of 100Hz, which amounts to around 90,000 samples for each IMU sensor for each position in the vehicle.

Of the trained models, the Decision Tree classifier performs well for most IMU sensor combinations. Furthermore, the gyroscope data alone gave the best classification results across the board. For 5.0s durations of data collection, the Rule-Based classifier gave the best accuracy of 99.71%, and for 0.5s and 1.0s durations of data collection, the Decision Tree classifier gave the best accuracy of 99.80% and 99.36%, respectively. The Rule-Based and Decision Tree classifiers both are computationally fast, so they are good choices for a real-time approach. Therefore, we determine that the combination of 5.0s of gyroscope data applied to a Rule-Based classifier is a good choice for determining whether a smartphone is in a person's hand while in a vehicle.

The gyroscope is capable of accurately detecting whether a smartphone is in someone's hand due to the context of what it measures, the rotational rate of change. The rotational measurement of the gyroscope is sensitive to the noise from metabolic and biological interactions with the smartphone; for instance, a heart beat propagated through the held hand. By extension, the features we selected—such as the variation and skewness—measure the noise only present in the gyroscope readings when a person is interacting with the smartphone, assisting the classifier determine when a person is holding the smartphone. For this reason, we believe that the result is accurate and that the selected classifier is able to leverage features from the gyroscope readings for accurate classification.

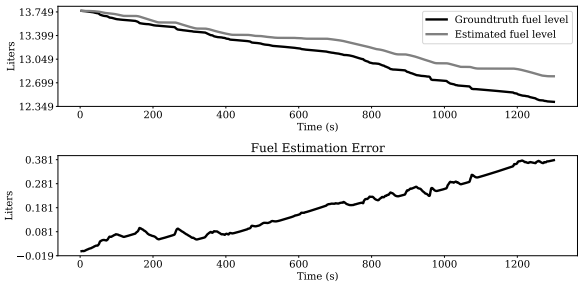
B. Accuracy of Vehicular Sensor Estimation

1) Evaluation Dataset: We evaluate CaRe using driving traces containing ground truth data from CAN and smartphone sensors. Our evaluation requires in-vehicle data that is beyond the scope of the OBD-II diagnostic standard, so we use OpenXC [60] to collect data. We collected data from 112 trips for 28.56 hours and 712.8 miles of driving in total. The trips cover highways and surface streets. We collected data from 7 different vehicles and 3 drivers, summarized in Tbl. III. The drivers placed the phone in a natural stationary location, such as the windshield, cup-holder or their pocket.

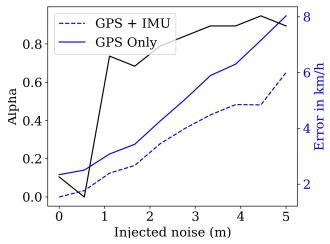
2) Estimation Accuracy: In what follows, we use the smartphone sensors to evaluate the estimation of the 5 in-vehicle sensors discussed in § V. We evaluate CaRe's estimation accuracy in the absence of anomalies by comparing the estimated and the ground truth values for all 112 trips. Plots of the CDF of the errors can be found in Fig. 3. The results presented in this section corroborate the estimation accuracy reported in [43], [62]. We have gone beyond prior work in our gear estimation evaluation and delved into the common cause of estimation failure for many sensors. Moreover, we evaluated the estimation algorithms under normal phone-usage scenarios. We note that due to the variation in smartphone sensor performance and specifications, the estimation accuracy may have differences.

Speed. CaRe can accurately estimate the vehicle speed using GPS sensors—50th percentile has $<$ 0.69 kilometers per hour (kmph) error, and 95th percentile has $<$ 4.74kmph error. The low-frequency speed ($<$ 1Hz) is accurately estimated using the GPS-inferred speed and the high-frequency speed ($>$ 1Hz) is estimated using the aligned accelerometer readings.

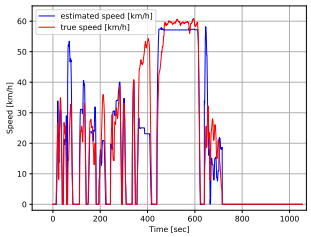
Gear. CaRe also accurately estimates the gear position. Over ≈80% of the time, it can exactly estimate the current gear position, and at 95% percentile, it is wrong by 1 gear position. We observed that the errors are related to where the driver travels. CaRe can more accurately estimate the gear position



(a) Fuel estimation error drift.



(b) Vehicle speed estimation error for increasing high-frequency GPS noise. For higher GPS noise, CaRe relies more on accelerometer component of the complementary filter.



(c) Speed estimation for a vehicle from the start to finish of a trip relying only on accelerometer. The Mean Average Error is 4.37 km/h. The red/blue line is the true/estimated speed.

Fig. 4: Sensor estimation error investigations.

when the trip is predominantly in the highway versus surface local roads. We divided the data from the highway and the local road by map-matching the GPS points. This effect occurs because the driver tends to stay in the same gear while on the highway, whereas there is much more fluctuation caused by start and stop behavior on the surface streets. This is further corroborated by the fact that gear estimation error and vehicle speed have a negative pairwise correlation of -0.1, meaning as the vehicle goes faster, there is less gear estimation error.

Steering Wheel Angle. CaRe can estimate steering wheel angle (SWA) with $< 2.02°$ error in 50% of all trips, and $< 38.2°$ in 95% of all trips. We find a strong negative correlation between SWA estimation and vehicle speed (coef=-0.22) and gear position (coef=-0.31). We find large errors in SWA when the car moves slowly. This occurs because as the car drives faster, there is a stronger relationship between the induced yaw rate in the vehicle body and the SWA.

Odometer & Fuel Level. CaRe can also accurately estimate odometer (50th% $<$ 0.15km, 95% $<$ 2.14km) and fuel level (50th% $<$ 0.09L, 95% $<$ 0.51L). Due to the accumulative nature of fuel estimation, we noticed an increasing error as the trip continues for a longer duration. The fuel-level estimate and drifting error are shown in Fig. 4a. The accumulating error only affects the estimates within a single trip. In between trips, CaRe re-calibrates the *change* in odometer and fuel level, and is therefore able to detect anomalies that happen during the trip.

3) GPS Noise: CaRe relies on GPS to estimate the vehicle speed. We evaluate the effect of GPS error/noise by adding normally-distributed noise to the GPS samples (sampled up to once every 100ms). We inject random noise for each 100ms time sample of the GPS signal. Realistic GPS noise is likely to be less jittery and noisy, but our analysis shows the effect of a more extreme noise distribution. We find that as we increase the high-frequency GPS noise, the average error also increases. The trend is that for every additional meter of high-frequency GPS noise, we see an average additional error of 1 kmph in the speed estimation using *only* the GPS. In fact, we find that with an increased GPS noise, CaRe relies more on the vehicle-aligned accelerometer to reduce the error. In Fig. 4b, this is shown as the α curve, the trade-off between GPS-speed and accelerometer-speed.

As seen in Fig. 4b, as the GPS noise gets too high, the speed estimations become increasingly inaccurate. We can use the GPS confidence returned by GPS chips to make use of rotation matrix techniques that do not make use of the GPS, such as those in [41], [42] and described in § IV-A. The results of such a method are plotted in Fig. 4c and are deemed to be acceptable compared to those that occur with high GPS noise.

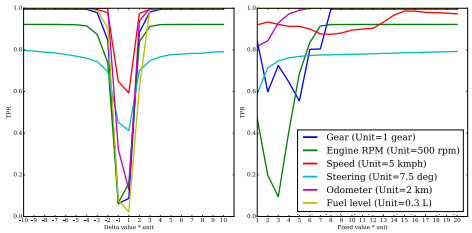


Fig. 5: The left figure shows TPR for varying injection magnitudes. The right figure shows varying *fixed* values of injection. At the moment of injection the vehicle sensor varies with the scenario, e.g., in some cases the car was traveling at 10 kmph when there was a 4kmph injection. For all cases, FPR was less than 5% and thus omitted.

C. Accuracy of Sensor-Falsification Detection

Next, we use the best settings learned from the estimation accuracy evaluation (§ VII-B) to evaluate the accuracy of sensor anomaly detection. This evaluation first involves injecting data into the CAN to mimic realistic anomalies (§ VII-C.1) and then detecting intrusion using CaRe (§ VII-C.2). We utilize the parameters found in § VI.

1) CAN Injection: We injected data into the collected data traces to simulate *sudden*, *gradual*, and *delta* anomalies. In the case of intentional anomalies (i.e., attacks), the adversary would resort to one of these three anomalies depending on the intended outcome. Some attack goals can be found in Tbl. I and in Tbl. IV.

In the evaluation of all three anomalies, we measure the number of flagged anomalies according to our model in § VI. We then determine the instances of true positive (TP), true negative (TN), false positive (FP), and false negative (FN). We used the normal case without injection to count FP and TN, and the injected case to count TP and FN. In what follows, we compare the True Positive Rate (TPR) and False Positive Rates (FPR) of various conditions. The TPR (also known as *recall*) is defined as $TP/(TP+FN)$ and the FPR is defined as $FP/(FP+TN)$. Configuring CaRe leads to a tradeoff between TPR and FPR. A cautious driver would prefer to increase TPR at the risk of more alarms. However, if the FPR is too high, it will raise too many false alarms and might lead the driver to get annoyed and ignore even real anomalies.

2) Detection Accuracy: **Speed.** For *sudden* anomalies, CaRe can detect speed injection anomalies with TPR=97.33%, FPR=0.2%. For a *delta* anomalies, CaRe can detect the

	Sensor	Values	Rationale
Sudden/Grad.	Speed	{ 4, 10, 25, 50, 100 } kmph	Used in manipulating other ECUs, *e.g.*, [4].
	Steer. Wheel Angle	{ -100, -50, -10, 10, 50, 100 } degrees	Trick adaptive headlight into shining in the wrong location.
	Gear	{ -1, 1, 4, 6 } gear	Used in manipulating other ECUs, *e.g.*, [4].
	Fuel level	{ 15.4 } gallons	Trick driver into emptying gas tank, *e.g.*, [3].
Delta	Speed	{ -50, -10, +10 } kmph	Trick driver into going faster, *e.g.*, [3].
	Steer. Wheel Angle	{ +10, +50 } degrees	
	Odometer	{ -1000 } km	Trick driver into missing oil change dates.
	Fuel Level	{ +0.5 } gallons	Trick driver into driving without sufficient fuel level, *e.g.*, [3]

TABLE IV: Specific injection values we used. The first 4 attacks were sudden or gradual injections. The last 4 were injected immediately as a delta of the actual value. These injections were inspired by existing literature.

injection with TPR=99.67% and FPR=0.2%. However, for *gradual* anomalies on the speed sensor, the accuracy drops to TPR=90.25% at FPR=0.2%.

Gear. CaRe can accurately detect any gear injection anomalies, with sudden anomalies at TPR=90.08%, FPR=0.22% and gradual anomalies at TPR=88.3%, FPR=4.79%.

Steering Wheel Angle. CaRe is able to detect SWA delta anomalies at TPR=91.98%, FPR=0%. The sudden anomalies are detectable with TPR=94.12%, FPR=0%. The gradual anomalies have a TPR=58.33%, FPR=0%. The primary reason why the gradual attack detection is less accurate for the SWA is the inherent difficulty of estimating the SWA using IMU sensors. This problem mainly occurs when the vehicle is traveling very slowly or stationary, *i.e.*, the driver may move the steering wheel significantly, but no vehicular dynamics occur. Therefore, to improve the TPR, we have to accommodate a much higher FPR, which may not be acceptable for drivers.

Fuel Level. We find a similar pattern in the fuel-level anomalies. CaRe can detect the delta anomalies with a TPR=93.86%, FPR=0.77%, and the sudden anomalies with TPR=88.8%, FPR=0.77%. However, it only detects the gradual anomalies with TPR=64.57%, FPR=0.77%. The poor performance of gradual anomaly detection is due to injected values closely resembling the actual values during the start of the anomaly. Only after a few seconds of the 10-second gradual anomaly does the value start to differ enough.

3) Weakest Detectable Anomaly: We evaluate the *minimum anomaly* CaRe can detect by injecting different magnitudes from the actual value for each sensor (Fig. 5). The x-axis is the magnitude of the sensor injection, which is multiplied by the unit scale shown in the figure legend. We can detect speed injection once it exceeds 10kmph. The detection threshold for other sensors are: Gear=2, Steering=22.5°, Odometer=4km and Fuel Level=0.9L.

In the left figure, we show varying magnitudes of a *delta* anomaly. As the anomaly is close to 0 delta (*i.e.*, the true value), the TPR drops to 0. In the right figure, we show the TPR for a sudden anomaly. If the sudden anomaly is close to the actual value, then the TPR becomes very low.

VIII. Discussion

Expected Deployment Scenario. Thanks to the ease of deployability due to the pervasive use of smartphones, CaRe has the potential for a wide reach. If an anomaly is detected, CaRe sends an alert of it, either directly to the driver or to another trusted agent such as a mechanic. This model resembles that of health apps, which are able to automatically send health data to the physician with the permission of the smartphone owner. Similarly, the alert of anomalies may aid a professional mechanic in diagnosing vehicle problems before they cause irreparable harm.

Following an alert of an anomaly, the driver/mechanic should carefully consider the context of the alert. If it is isolated with no other indication of an issue, then it is possible to be a false-positive. If it is indeed a false-positive, then this information can inform further iterations of refining CaRe. However, a stream of (or repeated) anomalies requires a careful inspection. Such a stream may indicate a malfunctioning vehicular sensor, an adversarial attack, or a false-positive invoked via an unexpected behavior. In this case, the driver may wish to exercise caution drive home slowly. They may also elect to have their vehicle enter its "limp home" mode, which is where an ECU triggers for the engine to prevent itself from driving faster. Thus, all cases provide useful feedback to the driver for enhancing both current and future vehicle safety and reliability.

Future Application. Additionally, smart infotainment OSs are on the horizon, such as Android Automotive OS (AAOS) [63]. AAOS has the ability to intelligently control various aspects of the vehicle, such as the HVAC and lighting systems. In the future, a smart infotainment OS such as AAOS may be coupled with CaRe in order to provide free sensor redundancy to any decisions that AAOS makes.

Incorporating Engine RPM. In our experiments, we considered estimating the engine RPM using speed, the final drive ratio, the transmission gear ratio, and tire circumference [62]. Figs. 5 contains some of the engine RPM evaluation. Our results demonstrated weak estimation performance with {50th, 95th}={115.6, 788.5} RPM error. This lead to weak detection performance for sudden (TPR=78.29%, FPR=0.54%) and gradual (TPR=43.22%, FPR=0.54%) anomalies. Tire slip causes this large variation in the engine RPM estimation, and is mediated by the tire slip ratio [19].

However, incorporating tire slip into the estimations is not simple. Several unpredictable factors cause tire slip such as the road condition, wear of the tire, and the friction coefficient between the tire and the road. We are able to show some factors can be considered. For instance, when we separated the estimation error based on the acceleration of the vehicle, we found that compared to normal acceleration conditions there was 1790% increase during sudden acceleration. Through a similar separation, we found 372% increased error in up-hill versus flat roads and 178% increased error in high-precipitation areas. Using these detectable factors can slightly improve the RPM estimation performance, but there are several challenges that remain.

IX. Acknowledgements

The work in this paper was supported in part by ONR under Grant No. N00014-22-1-2622. We thank the reviewers and RTCL for their feedback. Noah and Arun extend a special thanks to their families for lending their cars to the data collection efforts.

References

[1] I. Rouf, R. Miller, H. Mustafa, T. Taylor, S. Oh, W. Xu, M. Gruteser, W. Trappe, and I. Seskar, "Security and privacy vulnerabilities of in-car wireless networks: A tire pressure monitoring system case study," in *USENIX Security Symposium*, 2010.

[2] S. Checkoway, D. McCoy, B. Kantor, D. Anderson, H. Shacham, S. Savage, K. Koscher, A. Czeskis, F. Roesner, and T. Kohno, "Comprehensive experimental analyses of automotive attack surfaces," in *USENIX Security Symposium*, 2011.

[3] K. Koscher, A. Czeskis, F. Roesner, S. Patel, T. Kohno, S. Checkoway, D. McCoy, B. Kantor, D. Anderson, H. Shacham, and S. Savage, "Experimental analysis of a modern automobile," in *IEEE Symposium on Security & Privacy (S&P)*, 2010.

[4] C. Miller and C. Valasek, "Adventures in automotive networks and control units," in *DEF CON*, 2013.

[5] ——, "A survey of remote automotive attack surfaces," in *DEF CON*, 2014.

[6] ——, "Remote exploitation of an unaltered passenger vehicle," in *Black Hat USA*, 2015.

[7] K.-T. Cho and K. G. Shin, "Viden: Attacker identification on in-vehicle networks," in *ACM SIGSAC Conference on Computer and Communications Security (CCS)*, 2017.

[8] ——, "Fingerprinting electronic control units for vehicle intrusion detection," in *USENIX Security Symposium*, 2016.

[9] W. Choi, K. Joo, H. J. Jo, M. C. Park, and D. H. Lee, "VoltageIDS: Low-level communication characteristics for automotive intrusion detection system," *IEEE Transactions on Information Forensics and Security*, vol. 13, no. 8, 2018.

[10] R. Bhatia, V. Kumar, K. Serag, Z. B. Celik, M. Payer, and D. Xu, "Evading voltage-based intrusion detection on automotive CAN," in *The Network and Distributed System Security Symposium (NDSS)*, 2021.

[11] M. Kneib and C. Huth, "Scission: Signal characteristic-based sender identification and intrusion detection in automotive networks," in *ACM SIGSAC Conference on Computer and Communications Security (CCS)*, 2018.

[12] M. Kneib, O. Schell, and C. Huth, "EASI: Edge-based sender identification on resource-constrained platforms for automotive networks," in *The Network and Distributed System Security Symposium (NDSS)*, 2020.

[13] M. Foruhandeh, Y. Man, R. Gerdes, M. Li, and T. Chantem, "SIMPLE: single-frame based physical layer identification for intrusion detection and prevention on in-vehicle networks," in *Annual Computer Security Applications Conference (ACSAC)*, 2019.

[14] M. R. Moore, R. A. Bridges, F. L. Combs, M. S. Starr, and S. J. Prowell, "Modeling inter-signal arrival times for accurate detection of CAN bus signal injection attacks: A data-driven approach to in-vehicle intrusion detection," in *Conference on Cyber and Information Security Research (CISRC)*, 2017.

[15] Z. Tyree, R. A. Bridges, F. L. Combs, and M. R. Moore, "Exploiting the shape of CAN data for in-vehicle intrusion detection," in *IEEE Vehicular Technology Conference (VTC-Fall)*, 2018.

[16] M. Müter and N. Asaj, "Entropy-based anomaly detection for in-vehicle networks," in *IEEE Intelligent Vehicles Symposium (IV)*, 2011.

[17] M. Müter, A. Groll, and F. C. Freiling, "A structured approach to anomaly detection for in-vehicle networks," in *International Conference on Information Assurance and Security (IAS)*, 2010.

[18] A. Ganesan, J. Rao, and K. G. Shin, "Exploiting consistency among heterogeneous sensors for vehicle anomaly detection," SAE Technical Paper, Tech. Rep., 2017.

[19] K.-T. Cho, K. G. Shin, and T. Park, "CPS approach to checking norm operation of a brake-by-wire system," in *ACM/IEEE International Conference on Cyber-Physical Systems (ICCPS)*, 2015.

[20] A. Wasicek and A. Weimerskirch, "Recognizing manipulated electronic control units," SAE Technical Paper, Tech. Rep., 2015.

[21] M. Markovitz and A. Wool, "Field classification, modeling and anomaly detection in unknown can bus networks," *Vehicular Communications*, vol. 9, 2017.

[22] M. Marchetti and D. Stabili, "Anomaly detection of CAN bus messages through analysis of ID sequences," in *IEEE Intelligent Vehicles Symposium (IV)*, 2017.

[23] L. Xue, Y. Liu, T. Li, K. Zhao, J. Li, L. Yu, X. Luo, Y. Zhou, and G. Gu, "Said: State-aware defense against injection attacks on in-vehicle network," in *USENIX Security Symposium*, 2022.

[24] S.-F. Lokman, A. T. Othman, and M.-H. Abu-Bakar, "Intrusion detection system for automotive controller area network (can) bus system: a review," *EURASIP Journal on Wireless Communications and Networking*, vol. 2019, no. 184, 2019.

[25] K.-T. Cho and K. G. Shin, "Error handling of in-vehicle networks makes them vulnerable," in *ACM SIGSAC Conference on Computer and Communications Security (CCS)*, 2016.

[26] C. Miller and C. Valasek, "Advanced CAN injection techniques for vehicle networks," in *Black Hat USA*, 2016.

[27] F. Guo, Z. Wang, S. Du, H. Li, H. Zhu, Q. Pei, Z. Cao, and J. Zhao, "Detecting vehicle anomaly in the edge via sensor consistency and frequency characteristic," *IEEE Transactions on Vehicular Technology*, vol. 68, no. 6, 2019.

[28] I. Foster, A. Prudhomme, K. Koscher, and S. Savage, "Fast and vulnerable: A story of telematic failures," in *USENIX Workshop on Offensive Technologies (WOOT)*, 2015.

[29] T. Eden, "A reverse engineered interface for the BMW i3 electric car," https://github.com/edent/BMW-i-Remote, 2021.

[30] C. Wang, Q. Wang, K. Ren, N. Cao, and W. Lou, "Towards secure and dependable storage services in cloud computing," *IEEE Transactions on Services Computing*, vol. 5, no. 2, pp. 220–232, 2011.

[31] K.-T. Cho, K. Shin, Y. S. Kim, and B.-H. Cha, "Off is not off: On the security of parked vehicles," in *IEEE Conference on Communications and Network Security (CNS)*, 2020.

[32] H. Wen, Q. A. Chen, and Z. Lin, "Plug-N-Pwned: Comprehensive vulnerability analysis of OBD-II dongles as a new over-the-air attack surface in automotive IoT," in *USENIX Security Symposium*, 2020.

[33] P. Faruki, A. Bharmal, V. Laxmi, M. S. Gaur, and M. Conti, "Android security: A survey of issues, malware penetration, and defenses," *IEEE Communications Surveys & Tutorials*, vol. 17, no. 2, 2015.

[34] T. Sharma and D. Rattan, "Malicious application detection in Android — a systematic literature review," *Computer Science Review*, vol. 40, 2021.

[35] J.-H. Hong, B. Margines, and A. K. Dey, "A smartphone-based sensing platform to model aggressive driving behaviors," in *SIGCHI Conference on Human Factors in Computing Systems (CHI)*, 2014.

[36] J. E. Meseguer, C. T. Calafate, J. C. Cano, and P. Manzoni, "Drivingstyles: A smartphone application to assess driver behavior," in *IEEE Symposium on Computers and Communications (ISCC)*, 2013.

[37] J. Eriksson, L. Girod, B. Hull, R. Newton, S. Madden, and H. Balakrishnan, "The pothole patrol: Using a mobile sensor network for road surface monitoring," in *International Conference on Mobile Systems, Applications, and Services (MobiSys)*, 2008.

[38] S. Hu, L. Su, H. Liu, H. Wang, and T. F. Abdelzaher, "SmartRoad: Smartphone-based crowd sensing for traffic regulator detection and identification," *ACM Transactions on Sensor Networks*, vol. 11, no. 4, 2015.

[39] V. Coric and M. Gruteser, "Crowdsensing maps of On-Street parking spaces," in *IEEE International Conference on Distributed Computing in Sensor Systems (DCOSS)*, 2013.

[40] Y. Wang, X. Liu, H. Wei, G. Forman, C. Chen, and Y. Zhu, "CrowdAtlas: Self-updating maps for cloud and personal use," in *Annual International Conference on Mobile Systems, Applications, and Services (MobiSys)*, 2013.

[41] P. Mohan, V. N. Padmanabhan, and R. Ramjee, "Nericell: Rich monitoring of road and traffic conditions using mobile smartphones," in *ACM Conference on Embedded Network Sensor Systems (SenSys)*, 2008.

[42] K. Li, M. Lu, F. Lu, Q. Lv, L. Shang, and D. Maksimovic, "Personalized driving behavior monitoring and analysis for emerging hybrid vehicles," in *International Conference on Pervasive Computing*, 2012.

[43] L. Liu, H. Li, J. Liu, C. Karatas, Y. Wang, M. Gruteser, Y. Chen, and R. P. Martin, "Bigroad: Scaling road data acquisition for dependable self-driving," in *International Conference on Mobile Systems, Applications, and Services (MobiSys)*, 2017.

[44] J. Wiese, T. S. Saponas, and A. J. B. Brush, "Phoneprioception: Enabling mobile phones to infer where they are kept," in *SIGCHI Conference on Human Factors in Computing Systems (CHI)*, 2013.

[45] Y. Wang, Y. J. Chen, J. Yang, M. Gruteser, R. P. Martin, H. Liu, L. Liu, and C. Karatas, "Determining driver phone use by exploiting smartphone integrated sensors," *IEEE Transactions on Mobile Computing*, vol. 15, no. 8, 2016.

[46] Y. Wang, J. Yan, H. Liu, Y. Chen, M. Gruteser, and R. P. Martin, "Sensing vehicle dynamics for determining driver phone use," in *International Conference on Mobile Systems, Applications, and Services (MobiSys)*, 2013.

[47] H. L. Chu, V. Raman, R. R. Choudhury, A. Kansal, and V. Bahl, "Poster: You driving? talk to you later," in *International Conference on Mobile Systems, Applications, and Services (MobiSys)*, 2011.

[48] W. Wu, S. Dasgupta, E. E. Ramirez, C. Peterson, and G. J. Norman, "Classification accuracies of physical activities using smartphone motion sensors," *Journal of Medical Internet Research*, vol. 14, no. 5, 2012.

[49] A. Anjum and M. U. Ilyas, "Activity recognition using smartphone sensors," in *Consumer Communications and Networking Conference (CCNC)*, 2013.

[50] O. D. Incel, M. Kose, and C. Ersoy, "A review and taxonomy of activity recognition on mobile phones," *BioNanoScience*, vol. 3, no. 2, 2013.

[51] H. Martín, A. M. Bernardos, J. Iglesias, and J. R. Casar, "Activity logging using lightweight classification techniques in mobile devices," *Personal and Ubiquitous Computing*, vol. 17, no. 4, 2013.

[52] M. Shoaib, H. Scholten, and P. J. M. Havinga, "Towards physical activity recognition using smartphone sensors," in *International Conference on Ubiquitous Intelligence and Computing and International Conference on Autonomic and Trusted Computing (UIC-ATC)*, 2013.

[53] M. Ehatisham-ul Haq, M. A. Azam, J. Loo, K. Shuang, S. Islam, U. Naeem, and Y. Amin, "Authentication of smartphone users based on activity recognition and mobile sensing," *Sensors*, vol. 17, no. 9, 2017.

[54] C. Torrence and G. P. Compo, "A practical guide to wavelet analysis," *Bulletin of the American Meteorological society*, vol. 79, no. 1, 1998.

[55] S. Sardy, P. Tseng, and A. Bruce, "Robust wavelet denoising," *IEEE Transactions on Signal Processing*, vol. 49, no. 6, 2001.

[56] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, and M. Isard, "TensorFlow: A system for Large-Scale machine learning," in *USENIX Symposium on Operating Systems Design and Implementation (OSDI)*, 2016.

[57] Wikipedia, "Haversine formula," https://en.wikipedia.org/wiki/Haversine_formula, 2021.

[58] OpenStreetMap, "OpenStreetMap foundation wiki," https://wiki.osmfoundation.org/, 2021.

[59] Ford, "2017 Escape Tech Specs," https://media.ford.com/content/dam/fordmedia/NorthAmerica/US/Events/17-LAAS/2017-ford-escape-tech-specs.pdf, 2021.

[60] ——, "OpenXC," http://openxcplatform.com/, 2021.

[61] M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, and I. H. Witten, "The WEKA data mining software: An update," *ACM SIGKDD Explorations Newsletter*, vol. 11, no. 1, 2009.

[62] I. Skog and P. Händel, "Indirect instantaneous car-fuel consumption measurements," *IEEE Transactions on Instrumentation and Measurement*, vol. 63, no. 12, 2014.

[63] Android, "Android automotive," https://source.android.com/docs/devices/automotive, 2022.