A First Look at Vehicle Data Collection via Smartphone Sensors

Michael Reininger*, Seth Miller*, Yanyan Zhuang*†, Justin Cappos*
*NYU Polytechnic School of Engineering
†University of British Columbia

***Abstract*—Smartphones serve as a technical interface to the outside world. These devices have embedded, on-board sensors (such as accelerometers, WiFi, and GPSes) that can provide valuable information for investigating users' needs and behavioral patterns. Similarly, computers that are embedded in vehicles are capable of collecting valuable sensor data that can be accessed by smartphones through the use of On-Board Diagnostics (OBD) sensors. This paper describes a prototype of a mobile computing platform that provides access to vehicles' sensors by using smartphones and tablets, without compromising these devices' security. Data such as speed, engine RPM, fuel consumption, GPS locations, etc. are collected from moving vehicles by using a WiFi On-Board Diagnostics (OBD) sensor, and then backhauled to a remote server for both real-time and offline analysis. We describe the design and implementation details of our platform, for which we developed a library for in-vehicle sensor access and created a non-relational database for scalable backend data storage. We propose that our data collection and visualization tools are useful for analyzing driving behaviors; we also discuss future applications, security, and privacy concerns specific to vehicular networks.**

***Keywords*—Smartphone sensors, vehicular networks, data visualization and analysis**

I. Introduction

Modern smartphones and tablets have powerful computing, communications, and sensing capabilities [1]. In addition to being capable of performing complex computing tasks and communicating with each other wirelessly, smartphones and tablets have a rich set of on-board sensors, such as accelerometers, gyroscopes, GPSs, and cameras. These sensors provide valuable information when investigating users' needs and behavioral patterns [2]. Automobiles, a dominant means of transportation for several decades, are also beginning to be equipped with on-board sensors. These sensors, which provide Internet connectivity and vehicle condition monitoring, form a small ecosystem that promises to enhance both road safety and travel comfort. If properly harnessed together, the on-board sensors in cars and embedded sensors in smartphones can objectively record information gathered from a car's perspective, to benefit other motorists. Research scientists and engineers can use this data to test hypotheses, improve driving regulations, and design new systems for handling traffic. The communication power of smartphones to deliver large-scale vehicular data provides unique opportunities for improving the quality of life in today's modern smart roads and cities.

We present a prototype of a mobile computing platform that uses smartphones and tablets to access vehicles' sensors without compromising the devices' security. Using this platform, we are able to collect real-time data from vehicles, such as speed, engine RPM, fuel consumption, GPS location, etc. This data can then be backhauled and displayed on a remote server for analysis.

We describe the design and implementation of our platform and discuss the challenges we encountered. We first developed a library to allow access to in-vehicle sensor data and then designed a non-relational database for scalable, backend data storage. In particular, we were able to overcome problems of intermittent device connectivity and rapid changes in vehicles' movement that created significant challenges as we collected data. We demonstrate that our data collection and visualization tools are useful for analyzing driving behavior as well as individual or aggregated vehicles' commute patterns.

Vehicular data collection is still in its early phases. In the future, sensor data collection will see a wider range of applications. For example, using similar data, researchers will be able to deduce valuable information by fusing various sensor outputs. Security and privacy issues are on-going topics in vehicular networks research. We expect that more mechanisms will emerge that are secure and preserve user privacy.

The rest of this paper is organized as follows. Section II gives an overview of the existing technologies in vehicular data collection and the corresponding applications. Sections III and IV describe the design and implementation of our platform. Section V discusses future applications along with security and privacy concerns in vehicular networks and Section VI concludes the paper.

II. Background and Motivation

A. Data Collection Techniques

Over the past decades, researchers have developed several techniques for collecting vehicular data on highways and in (sub)urban areas. Traditional approaches use road-mounted detectors such as cameras, microwave sensors, and bluetooth scanners, etc [3]. These technologies provide information about the number and type of vehicles passing through an area, their speed, travel time, and so on. However, such approaches are complex and involve high installation costs [4]. For example, sensors need to connect to a vehicle's third-party, custom onboard computer in order to receive signals from these sensors. In some cases, engineers need to adjust the GPS antenna and install equipment in one of the car doors or on the roof. These factors have lead to limited network coverage.

Recently, standards organizations have allocated new wireless communication channels specifically designed for auto-

motive use [5][1]. This has enabled new technologies such that automobile systems can now rely on On-Board Units (OBUs) to report vehicle information directly to other vehicles or to a central server. With sophisticated on-board computing units installed in vehicles, researchers are able to collect measurement data to monitor traffic [6], detect traffic accidents and propagate emergency messages [7], communicate diagnostic information [8], and so on. However, a major limitation of this approach is the availability of OBUs. It is likely that such an advanced technology is only available to a biased subset of vehicles.

More recently, the rising use of cellular technologies has made vehicle-based data collection much more convenient. One new technique, called floating cellular data (FCD) [9], has been proposed by researchers [10]. This technique uses anonymized cellular network data from mobile phones of drivers or passengers inside vehicles. However, this approach requires complex location triangulation and access to data stored by a network operator. Although the technique does not require on-road infrastructure, a monitoring system that intercepts cellular network communications and parses signaling protocols is necessary.

For our research, we use everyday Android smartphones and WiFi On-Board Diagnostics sensors in order to collect vehicular sensor data and backhaul the data to a database for visualization and analysis. To be able to collect data from vehicles, no expert knowledge about the hardware or protocol configuration of a vehicle's OBU is needed. Moreover, it is not necessary to make modifications to the existing network communication infrastructure or to mobile devices.

B. Example Use Cases

Current research in vehicular telematics and intelligent transportation systems (ITS) emphasizes accident prevention platforms, infotainment systems, and the discovery and analysis of eco-friendly routes [11]. Vehicular network applications can help drivers access current traffic conditions, improve transportation safety and efficiency, and provide such services as travel planning and teleconferencing to both drivers and passengers, while on the move.

Additional research on automotive data has helped develop platforms for analyzing driving behavior. These platforms have been increasingly centered around smartphones [12], [13], [14]. Indeed, smartphone apps can now understand driving behaviors using built-in sensors like the accelerometer, gyroscope, and GPS [15]. Furthermore, they can alert drivers to potential road hazards based on driving styles and patterns. Although our prototype also focuses on smartphones and their ability to collect environmental data, our work is novel in that we provide a platform for resulting applications to collect, analyze, and visualize both car data (e.g. fuel consumption and pressure, mileage, engine RPM, etc.) and smartphone data in a streamlined yet secure fashion, from an anonymous, voluntary user base. This allows applications to analyze not only one, but two sources of data, in real-time, and to share that information with drivers. As this research evolves, we envision this platform offering feedback to drivers on driving behaviors, traffic information, road conditions, potentially unsafe situations, and eco-friendly routes. All of this information can be shared in the cloud across drivers' smartphones, arriving in the form of alarms from servers in the cloud, or in the form of peer-to-peer alerts from neighboring vehicles.

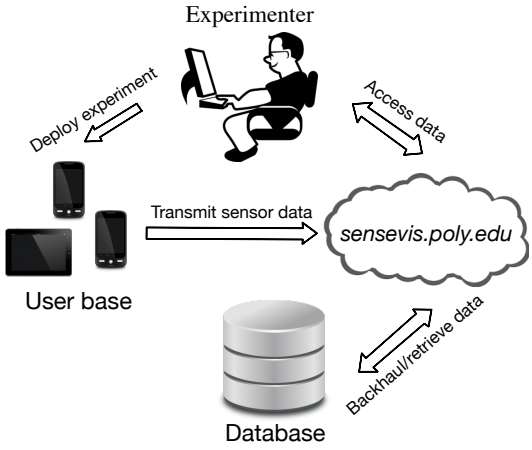

Fig. 1: The process of collecting, storing, and visualizing sensor data.

III. Vehicular Data Collection Design

Our vehicular data collection consists of a backend that directly communicates with in-vehicle sensors and collects sensor data. This backend is part of a distributed testbed that allows access to smartphone sensor data in a secure and performance-isolated way [2]. Furthermore, data can be backhauled to a centralized server for permanent data storage. This server also serves as a frontend to make the data available to experimenters interested in traffic monitoring, etc. In this section we describe the design of our platform.

A. Data Collection Infrastructure

1) Distributed Smartphone Sensor Testbed: We deploy our platform on a distributed testbed [2] that provides an efficient programming environment for our data collection. Experimenters can implement automated experiments and collect data from accelerometer, GPS, WiFi, camera, and other sensors on mobile devices that are owned by the general public. Conducting these experiments requires minimal power and networking resources, used in a non-intrusive manner. Our testbed employs a light-weight sandbox [16] to limit the amount of storage, network, memory, battery, and CPU resources used on a mobile device. To protect end users' devices from malicious attackers, our sandboxed programs are securely isolated from other programs on the same device.

2) Data Collection Overview: The process of deploying an experiment on the testbed and then collecting data from vehicular sensors on end user devices is as follows. *Device owners* install a testbed app [17] on their devices[2]. Since we are interested in vehicular data, the target group of device owners are also vehicle owners. These owners simply insert their WiFi On-Board Diagnostics (OBD) [18] sensor into

[1]This standard is called DSRC (Dedicated Short-Range Communications).

[2]Currently, Android smartphones and tablets are supported.

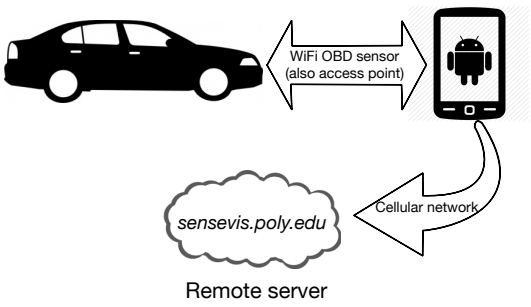

Fig. 2: Design of `obdlib`.

their cars' OBD ports (located under the steering wheel), and connect their smartphone or tablet to the sensor, which also runs as a WiFi access point. Note that OBD systems are in most cars and light trucks on the road today [19]. Therefore, our infrastructure does not require extra installation of specialized hardware or equipment.

An *experimenter* interested in getting vehicular data from device owners' smartphones and tablets will

1) Register with our testbed infrastructure and acquire end user devices for experimentation.
2) Write experiment code using the sandboxed programming environment [16].
3) Upload code to a set of smartphone devices that she or he has access to (obtained in step 1), using our testbed's experiment management tool [20], .
4) Start or stop the experiment at anytime; the experimenter can also collect sensor data using the testbed's logging function.

The experiment code runs in the background of those mobile devices. Therefore, device owners need not worry about having to interact with the app through an interface or about being interrupted as they use their smartphones. Concurrently and inconspicuously, the background experiment records sensor data and can backhaul the collected data to a remote server.

At the remote server, the vehicular sensor data collected from multiple end user devices is stored in a non-relational database (our reason for using a non-relational database is stated in the following section). The collected data set can be visualized on Google Maps to identify fuel efficient routes, routes with higher traffic activity, and routes (and drivers) that exhibit a high frequency of reckless or illegal driving behaviors, etc. Figure 1 summarizes the process of collecting, storing, and visualizing sensor data.

B. Vehicle Data Collection

We designed a library for the sandbox that provides an interface to communicate with an in-vehicle OBD sensor, which also acts as a WiFi access point. We named the OBD communication library `obdlib` [21] and detail its implementation in Section IV-A. In our prototype implementation, the experiment code can connect to the ODB sensor, similar to connecting to a WiFi hotspot, and collect vehicle data from this sensor through the `obdlib` interface. Our `obdlib` allows

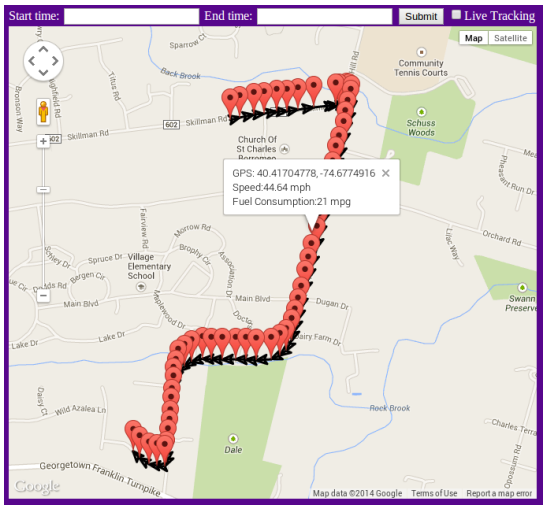

Fig. 3: An example of vehicular data.

access to vehicular sensor data such as speed, engine RPM, and rate of fuel consumption. We implemented the experiment such that the data collected is periodically cached on the phone and backhauled to a remote server whenever an Internet connection is available through the cellular network. The design of the communication protocol across the vehicle, the smartphone, and the remote server is shown in Figure 2.

The rationale for having a centralized server to collect vehicular sensor data is to support data analytics and visualization. In addition to uploading the data to the server, an experimenter can take advantage of our distributed testbed to share vehicular sensor data in a peer-to-peer fashion with other experimenters.

C. Frontend: Data Backhaul and Visualization

To handle the storage of multiple Android devices' vehicular sensor data in a user-friendly way, we deployed a website [22] that accepts `POST` requests that contain the desired (and encrypted) sensor data and stores it under private experimenter accounts. Experimenters, however, need not construct the requests nor encrypt sensor data themselves, because we provide a library, named `storesense`, to execute both of these functions [23]. With a single function call from `storesense`, an experimenter can securely backhaul their sensor data into a database on our website's server. Thereafter, experimenters can log into their accounts on our website to analyze, visualize (in tables, by default), and interact with the data they collect from remote end users' smartphones.

Figure 3 shows a visualization of a subset of our vehicular sensor data displayed on a map using the Google Maps API [24]. This map feature is built into our website. In the figure, a trajectory of automobile locations, spanning a five minute period from a test run on August 5, 2014, is shown on a map. By clicking on each marker in Figure 3, an experimenter can observe other sensor data collected, along with the vehicle's GPS location, such as the speed and fuel consumption of the vehicle at different times. Using our data set, an experimenter can observe fuel efficiency as it varies

```
{
  "sensors": {
    "speed_car": 60,
    "maf_car": 4,
    "rpm_car": 114,
    "gps_phone": {
      "error": null,
      "id": 0,
      "result": {
        "network": {
          "time": 1407200660927,
          "speed": 60,
          "altitude": 4.099999904632568,
          "bearing": 82.6999694824219,
          "provider": "gps",
          "longitude": -73.986706,
          "latitude":40.694010,
          "accuracy": 7,
        }
      }
    },
    "id": "310410696731709",
    "time": "ISODate("2014-08-05T21:04:07.183-04:00")
}
```


Fig. 4: JSON document of vehicular data.

across different routes, traffic densities, and speeds, or whether a specific vehicle exhibits reckless or illegal driving behaviors.

In Figure 3, each marker on the map is separated by five seconds. This sampling rate was pre-set in our implementation. In the future, however, sampling rates will be automatically adjusted via a mechanism called a security layer [16]. Such a security layer acts as a protection between program interfaces and experiment code, and protects functionality from tampering [25], [26]. To do this, we will implement a security layer that prevents the experiment code from sampling sensor data too frequently or reduces the sensor data granularity. This is required for a number of reasons. On the one hand, we need to limit the frequency of data sampling to avoid battery drain on users' devices. On the other hand, sensor data can be filtered in response to user requests, thus preserving their privacy. For example, an experiment could be prevented from obtaining the precise GPS location of a vehicle; instead the data is abstracted to show only neighborhood or zip code information.

IV. Vehicular Data Collection: Prototype Implementation

In this section, we describe the implementation of the prototype we described in Section III.

A. Vehicular Data Collection

For our prototype platform, the OBD-II ELM327 WiFi sensor was used. In order to facilitate communications between sandboxed programs on a smartphone and the OBD sensor, we developed the `obdlib` library [21], which implements the standard OBD communication protocol. We used a TCP connection between the smartphone and the OBD sensor via the default WiFi gateway IP address assigned to the OBD sensor[3].

In addition to the TCP connection wrapper, `obdlib` includes OBD standard parameter ID (PID) commands that

[3]By default, our OBD sensor listens on port 35000 at address `192.168.0.10`.

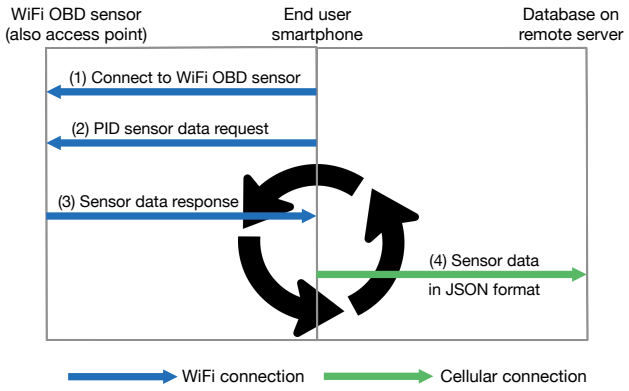

Fig. 5: State diagram for our data collection prototype.

can be used to request a specific data element from the OBD sensor [27]. After getting the data, an experimenter again uses `obdlib` to decode the response from the sensor. Vehicular sensor data is first logged onto the smartphone and structured in JSON format. An example of this data is shown in Figure 4.

In order to give experimenters access to collected data and to save storage space on users' mobile devices, the data should be moved to a reliable server. However, smartphone network connections are very unreliable because a device owner may move between different networks or disconnect from the Internet at any time. We overcome the problems of intermittent network connectivity by caching JSON sensor data samples into a buffer and sending them opportunistically whenever Internet connectivity is available. Figure 5 shows the connection state diagram for our data collection prototype.

B. Non-relational Database Store

Given the variety of sensors on a smartphone, sensor measurements come in different forms. As shown in Figure 4, many types of sensor data are far more complex than are simple primitive data types. For example, GPS data is returned to an experimenter as a JSON object that contains another "result" JSON object. This object, in turn, contains yet another "network" JSON object, and this is the final container of recognizable quantities such as latitude, longitude, altitude, and so on.

As a result, it can be very challenging to store sensor data in a relational database, which has a fixed column structure. The use of a relational database would require either the creation of tables with as many columns as sensors or the creation of tables upon request by experimenters, who would have to predetermine which sensors they intend to use. The former option is not reasonable—for every unused sensor, these massive tables would waste space in the database. While the latter option is plausible, it places a burden on experimenters to know the data types of all the sensor data they plan to collect *before* their experiment begins. If an experimenter wants to use different sensors later — for example, when new sensors are available on next generation devices — each database table would have to be copied over into a new table that could accommodate these additional sensors.

For these reasons we decided to use a non-relational database, MongoDB [28], for storing sensor data. MongoDB has a Binary JSON (BSON) document-style structure identical to that shown previously in Figure 4, albeit converted into binary as its name suggests. This provides the scalability that is crucial to our system and that allows dynamic storage of new sensors, as needed. Non-primitive data types can also be stored more easily because no column structure exists in MongoDB that enforces model schema or data typing. Additionally, because our sensor data is already in JSON format, MongoDB is the most convenient storage option. For the purposes of securing stored sensor data and easing the flow of an experiment, this implementation is made transparent to the user by our website's interface.

Within the database, individual devices are distinguishable to an experimenter by using a stored, unique identifier. We chose to use the device's IMEI number because it is a 15 digit ID that is unique to each device's hardware [29]. Although the SIM number might be a more familiar choice, we decided not to use it because the devices used in our testbed are not required to have data plans. Experimenters can instead opt to run their experiments solely through WiFi networks.

V. Smartphone Vehicle Sensors: A Look to the Future

In the future, we hope that smartphone-based vehicular telematics platforms will enable researchers to contribute to a greener planet, a decrease in traffic related accidents, and ultimately, serve as a platform that could save lives. Meanwhile, security and privacy concerns surrounding smartphone-based vehicular systems need to be addressed. We discuss these issues and our design choices in this section.

A. Future Work With Smartphone Sensors

We envision the future of smartphone-based vehicular telematics as a platform that will fuse data from multiple sensor sources, and thus, allow us to make inferences using exogenous variables. For example, weather information or hotspot traffic caused by large gatherings at entertainment venues could be correlated with vehicular sensor data to predict possible hazardous road conditions. With the multitude of available sensors embedded in smartphones and on vehicular OBUs, we expect that sensors will assist drivers in making decisions. Additionally, transportation authorities could use changes in aggregated gas mileage information—associated with specific stretches of road within municipalities—to prioritize road repairs, re-calibrate traffic lights, and dynamically adjust speed limits in order to support smart city goals.

Additionally, context-aware data provided by smartphones could help us understand real-time transit patterns. For example, we can deduce present road conditions and avert potential road hazards based on the frequency that specific smartphones pass each other, both those that belong to emergency response teams and to the general public. This information should be provided by the smartphone and by the built-in infotainment systems that come with most new cars. This will promote distraction-free driving.

B. Data Privacy and Security

As with most technologies, smartphone-based vehicular systems could be a double-edged sword if not properly used. For example, if the data collected is not anonymized, it could be linked to individual users, which would pose serious surveillance threats if the smartphones were connected to a server that was accessible by third parties [30]. We are currently developing tools that will blur sensitive information [25], [26] and effectively preserve user privacy.

As with other technologies, smartphones and sensors may be targets of malicious actions by third parties. Therefore, protective measures must be taken to ensure the security of in-vehicle smartphones and sensor systems. The system must guarantee that even if a hacker circumvents the sensor environment, no harm can be done to users' smartphones and more importantly, their vehicles' systems. These factors must be taken into account during the design stage of smartphone-based vehicular telematic systems. In the prototype we describe in this paper, the sandboxing mechanism in Section III-A effectively prevents malicious attacks. We expect that more secure mechanisms will appear in the future.

VI. Conclusion

In this paper, we describe a prototype of a smartphone-based platform that provides access to vehicular data such as speed, engine RPM, fuel consumption, GPS locations, etc. In our platform, data can be collected from moving vehicles by using smartphones in combination with On-Board Diagnostics sensors and an in-vehicle sensor library we designed. Collected data can be backhauled to a remote server for real-time and offline analysis. In describing the rationale for our system design and the challenges we encountered, we demonstrate that our data collection, backhaul, and visualization methods are useful for analyzing driving behavior. Based on this work, we consider future applications, security, and privacy concerns that surround vehicular networks. Our work presents a first look at vehicular data collection via smartphone sensors, as well as their potential both today and in the future.

Acknowledgment

This material is based upon work supported by the National Science Foundation under Grant No. 1205415. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation. This work was supported in part by American International Group Inc.

References

[1] L. Cai, S. Machiraju, and H. Chen, "Defending against sensor-sniffing attacks on mobile phones," in *Proceedings of the 1st ACM workshop on Networking, systems, and applications for mobile handhelds*. ACM, 2009, pp. 31–36.

[2] "Sensibility Testbed," https://sensibilitytestbed.com/.

[3] J. Zhang, F.-Y. Wang, K. Wang, W.-H. Lin, X. Xu, and C. Chen, "Data-driven intelligent transportation systems: A survey," *Intelligent Transportation Systems, IEEE Transactions on*, vol. 12, no. 4, pp. 1624–1639, 2011.

[4] "System Installation & Data Collection," http://www.infodev.ca/vehicles/vehicle-installations-data-collection.html.

[5] "Dedicated Short Range Communications (DSRC) Home," http://en.wikipedia.org/wiki/Dedicated_short-range_communications.

[6] Y. Zhuang, J. Pan, V. Viswanathan, and L. Cai, "On the uplink mac performance of a drive-thru internet," *Vehicular Technology, IEEE Transactions on*, vol. 61, no. 4, pp. 1925–1935, 2012.

[7] Y. Zhuang, J. Pan, Y. Luo, and L. Cai, "Time and location-critical emergency message dissemination for vehicular ad-hoc networks," *IEEE Journal on Selected Areas in Communications*, vol. 29, no. 1, pp. 187–196, 2011.

[8] Y. Zhuang, J. Pan, and L. Cai, "A probabilistic model for message propagation in two-dimensional vehicular ad-hoc networks," in *Proceedings of the seventh ACM international workshop on VehiculAr InterNETworking*. ACM, 2010, pp. 31–40.

[9] "Floating car data," http://en.wikipedia.org/wiki/Floating_car_data.

[10] A. Janecek, K. A. Hummel, D. Valerio, F. Ricciato, and H. Hlavacs, "Cellular data meet vehicular traffic theory: location area updates and cell transitions for travel time estimation," in *Proceedings of the 2012 ACM Conference on Ubiquitous Computing*. ACM, 2012, pp. 361–370.

[11] P. Papadimitratos, A. La Fortelle, K. Evenssen, R. Brignolo, and S. Cosenza, "Vehicular communication systems: Enabling technologies, applications, and future outlook on intelligent transportation," *Communications Magazine, IEEE*, vol. 47, no. 11, pp. 84–95, 2009.

[12] "PhoneLab: Programmable Participatory Smartphone Testbed," http://www.phone-lab.org/.

[13] "Mobile Territorial Lab (MTL)," http://www.mobileterritoriallab.eu/.

[14] G. Larkou, C. Costa, P. G. Andreou, A. Konstantinidis, and D. Zeinalipour-Yazti, "Managing smartphone testbeds with smartlab," in *Presented as part of the 27th Large Installation System Administration Conference*. USENIX, 2013, pp. 115–132.

[15] "Apps for monitoring teen drivers," http://blog.cochran.com/wordpress/index.php/apps-monitor-teen-drivers.

[16] J. Cappos, A. Dadgar, J. Rasley, J. Samuel, I. Beschastnikh, C. Barsan, A. Krishnamurthy, and T. Anderson, "Retaining sandbox containment despite bugs in privileged memory-safe code," in *Proceedings of the 17th ACM conference on Computer and communications security*, ser. CCS '10. New York, NY, USA: ACM, 2010, pp. 212–223. [Online]. Available: http://doi.acm.org/10.1145/1866307.1866332

[17] "Sensibility Testbed, Google Play Store," https://play.google.com/store/apps/details?id=com.sensibilitytestbed.

[18] "On-board diagnostics," http://en.wikipedia.org/wiki/On-board_diagnostics.

[19] "Does My Car Have OBD-II?" http://www.obdii.com/connector.html.

[20] "Seash: The Seattle Shell – Seattle – Trac," https://seattle.poly.edu/wiki/SeattleShell.

[21] "obdlib," https://github.com/CyberAdmin/obdlib.

[22] "Sensevis," http://sensevis.poly.edu/.

[23] "storesense," https://github.com/smmiller/sensevis/blob/master/storesense.r2py.

[24] "Google Maps API," https://developers.google.com/maps/.

[25] "BlurSense," https://blursense.poly.edu.

[26] J. Cappos, L. Wang, R. Weiss, Y. Yang, and Y. Zhuang, "Blursense: Dynamic fine-grained access control for smartphone privacy," in *Sensors Applications Symposium (SAS), 2014 IEEE*. IEEE, 2014, pp. 329–332.

[27] "OBD-II PIDs," http://en.wikipedia.org/wiki/OBD-II_PIDs.

[28] "MongoDB Architecture," http://www.mongodb.com/mongodb-architecture.

[29] "IMEI.info," http://www.imei.info/.

[30] "Data mining the new black box of self-driving cars," http://theconversation.com/data-mining-the-new-black-box-of-self-driving-cars-31685.