CaraoKey : Car States Sensing via the Ultra-Wideband Keyless Infrastructure

Avinash Kalyanaraman[†*], Yunze Zeng[‡*], Sushanta Rakshit[‡], Vivek Jain[‡]
[†]University of Virginia, Charlottesville, VA, USA; Email: ak3ka@virginia.edu
[‡]Bosch Research, Sunnyvale, CA, USA; Email:{yunze.zeng, sushanta.rakshit, vivek.jain}@us.bosch.com
[*]The first two authors contributed equally to this work.

***Abstract*—Automobile manufacturers are building cars with passive keyless entry support – the ability for users to unlock and start their cars while having the car keys "in their pockets". In this regard, car manufacturers are currently installing Ultra-Wideband (UWB) radios in cars owing to their robustness to relay attacks and their precise ranging estimates. In this paper, we explore the possibility of using these UWB radios (that exist for keyless entry) as an orthogonal sensing modality. To test this hypothesis, we build *CaraoKey*, the first system that exploits the UWB keyless infrastructure as a "sensor". More specifically, *CaraoKey* leverages the UWB radios to infer the state of a car (empty, door, window or trunk open, person inside the car, etc.). It does so by building a multipath profile based on the Channel Impulse Response (CIR) that is computed by each UWB node. We design and implement a 14-node system (a superset of all possible node locations) and evaluate it at 7 different indoor and outdoor locations while capturing a wide range of factors such as cars, people, walls on the sides, etc. Our results indicate that *CaraoKey* can distinguish 8 different car states with 98% accuracy using 8 nodes, and nearly 94% accuracy using just 4 nodes.**

I. Introduction

The keying system in automobiles has evolved significantly from the initial usage of a mechanical key. Car manufacturers have the passive keyless entry vision [1], [2] – a car automatically unlocks itself when the person (who carries a compatible device such as a key fob or smartphone) is in its vicinity, and the car can be started only when the device is inside the car. To realize this vision, the current state-of-the-art solution uses LF-UHF (a combination of low-frequency and ultra high frequency) channels. However, these systems can be subject to relay attacks [2], [3] – as recent as October 2018 [4]. Consequently, car manufacturers are now developing Ultra Wideband (UWB)-based solutions for keyless entry [5], [6], [7]. These UWB systems are more robust to relay attacks as the IEEE 802.15.4-2015 UWB standard [8] explicitly incorporates timing information. In fact, the IEEE 802.15.4z Enhanced Impulse Radio Task Group is currently tasked with developing more accurate ranging methods with UWB keyless access as one of its main pilot applications [9]. Given this impending installation of UWB radios for keyless entry, we ask the question if we could multi-purpose these UWB radios – i.e. leverage them for secondary use-cases beyond keyless entry. More specifically, we explore the possibility of using the UWB keyless infrastructure as a sensing modality.

Sensing systems that leverage an already existing infrastructure (e.g. Wi-Fi [10], [11], acoustic [12], visible light infrastructure [13], etc.) can be used in three main ways – (i) in a standalone manner that mitigates the need for extra hardware resulting in cost, space and/or power savings, or (ii) be used in combination with other sensing systems to improve data fidelity, or (iii) to trigger a power hungry or more privacy invasive sensing system like cameras. In this paper, we are the first to explore the UWB keyless infrastructure of automotives as a sensing modality. Such a sensing system that piggybacks on the existing UWB infrastructure can enable several applications(e.g. intrusion/activity detection and occupancy counting). We build *CaraoKey*, which explores the possibility of inferring car states using these UWB sensors.

CaraoKey performs UWB-based sensing of car states by leveraging the channel impulse response (CIR) computed by UWB receivers. As shown in Figure 1, the CIR is indicative of reflections in the environment, and changes as the state of the car changes. *CaraoKey* captures these CIR changes to identify the car state in two steps. First, it prunes the state space (identifies the most likely states) by correlating the CIRs observed by the receivers with a corpus of reference CIRs. Next, it narrows down the car state by computing a *multipath profile*, a measure of how the car is reflecting in each state as observed by the UWB nodes. Figure 2 shows an example of how two features contained in the *multipath profile* help separate some of the states. However, *CaraoKey* must deal with two key challenges while building the *multipath profile*. Firstly, the UWB transceivers in the car are not synchronized. Consequently, each CIR computed by a receiver will be randomly shifted with respect to previously computed CIRs from the same transmitter. *CaraoKey* addresses this challenge by identifying the first (direct) path in the CIR, and aligning the CIRs about this path, thus yielding a repeatable signature. Secondly, *CaraoKey* must be robust to changes in the location of the automobile (i.e. the same solution must work in a multipath rich indoor garage, a parking lot with cars on the sides, in free space, etc.). To address this challenge, *CaraoKey* leverages the internal UWB nodes to build the multipath profile which are more robust to location changes.

To test our hypothesis, we deploy 14 UWB nodes (placed

[†]Avinash Kalyanaraman completed this work during his internship at Bosch Research, Sunnyvale, CA, USA.

978-1-7281-6630-8/20/$31.00 2020 © IEEE

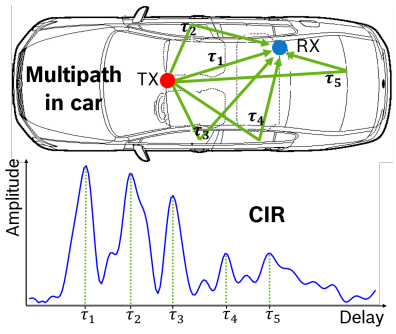

Fig. 1: The CIR is representative of the multipath reflections inside the car.

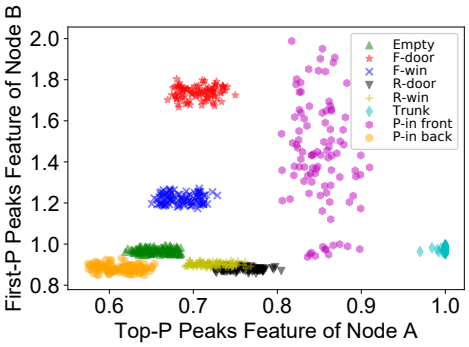

Fig. 2: An example of two features in the *multipath profile* that aid in state identification.

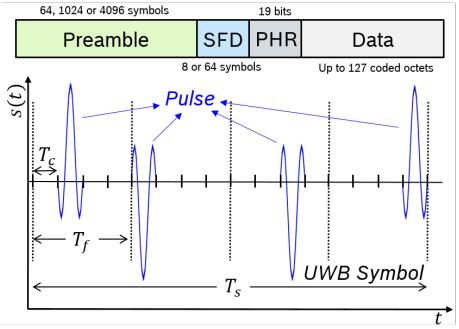

Fig. 3: UWB packet and UWB symbol structures.

inside and outside the car) in two different configurations of a sedan. These node locations are chosen from a superset of possible node locations [14], [15], [16]. This allows us to determine optimal number of nodes and their locations for *CaraoKey* which can further motivate manufacturers to install UWB nodes at those locations. A typical keyless solution may have up to 8 nodes depending on make/model and OEMs' preference, where up to 4 nodes may be external. We evaluate *CaraoKey* at 7 different physical locations – 2 indoor locations in a multipath rich garage, and 5 outdoor locations with cars, people, walls on the sides, etc. We show that using 8 UWB nodes, *CaraoKey* can distinguish between the 8 states of interest with an accuracy of 98%. In comparison, a system that is based on received signal strength (RSS) (e.g. Bluetooth), achieves only 49% accuracy. We then show that using only 4 nodes, *CaraoKey* can achieve a comparable accuracy of 94%.

In summary, the paper makes the following contributions:

- *CaraoKey* is the first system that leverages the UWB keyless infrastructure in automobiles as a sensing modality. As a first application, it demonstrates the sensing capability by inferring the state of a car.
- We show that the interior UWB nodes are more conducive for car interior sensing and more resistant to location changes than the exterior nodes. We identify these conducive UWB node locations for states sensing by building a node connectivity profile and performing receiver selection analysis on a multi-node real car testbed.
- *CaraoKey* is built in a manner that does not warrant any UWB node synchronization and is robust to location changes. It works via a two-step process of building a *multipath profile* – a measure of how the car reflects in each state. This profile is constructed from the CIR computed by UWB receivers.
- Upon prototyping on standards-compliant hardware and evaluating at 7 diverse indoor and outdoor locations, *CaraoKey* identifies the states of interest with 98% and 94% accuracy using 8 and 4 UWB nodes respectively.

II. Related Work

Sensing systems that leverage an existing infrastructure have interested the research community with numerous Wi-Fi [10], [11], [17](and other RF modalities [18]), acoustic [12] and visible-light based sensing systems [13]. There also exists systems that leverage these infrastructure in cars to sense context of recognizing gestures for hands-free control [19], detecting phone usages while driving [20], tracking the drivers head [21], sensing driver distractions [22], etc. Our work is different from these systems in that we use a different sensing modality, UWB, to infer car states. Comparing with other UWB sensing work of identifying materials and their quality [23], or counting people walking through doorways [24], *CaraoKey* performs UWB sensing in the context of automobiles by computing a *multipath profile* from the CIR computed by UWB nodes. To the best of our knowledge, ours is the first system that leverages UWB radios to infer car states.

III. Approach

Figure 4 provides an overview of *CaraoKey's* working. In this section, we start with a brief introduction of UWB background and then elaborate each step of *CaraoKey* in detail.

A. UWB Background

In *CaraoKey*, a transmitter periodically beacons (blinks) an UWB message. This blink message is in the IEEE 802.15.4 UWB format [8]. As shown in Figure 3, an UWB Packet contains header (preamble, start of frame delimiter, PHY header) and payload symbols. These UWB symbols travel over the air across multiple paths and reach the receiver UWB nodes (slaves). An UWB receiver uses the perfect periodic auto-correlation property of the known preamble sequence [25] to compute the impulse response of the channel. Said differently, it runs a correlator that correlates the received signal with the known preamble sequence to compute a channel impulse response (CIR) which is given by :

$$h(t) = \sum_{l=1}^{L} \alpha_k \delta(t - \tau_l) \quad (1)$$

where $\delta(.)$ refers to the Dirac delta function. *CaraoKey* uses this CIR which is indicative of the L reflected paths, to identify the car state by leveraging the intuition that the different states affect the CIR differently.

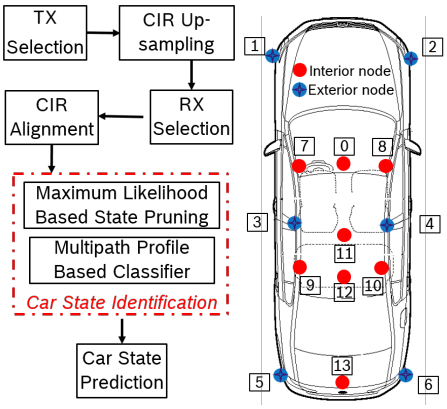

Fig. 4: System overview of *CaraoKey*

Fig. 5: Experiment node deployment.

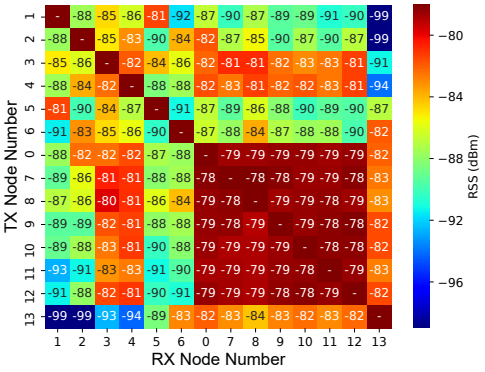

Fig. 6: The average RSS for an interior UWB transmission is higher and mostly symmetrical.

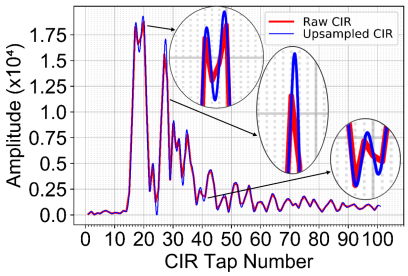

Fig. 7: CIR upsampling: Each receiver upsamples its CIR via an FFT.

B. Connectivity Test – Transmitter Selection

In order to identify node locations to perform sensing of car states via UWB nodes, we start from first principles. In other words, we start by over-instrumenting a car with 14 nodes, as shown in Figure 5, and then narrow down the node positions of interest for car-state sensing. These 14 nodes are distributed both inside and outside the car as UWB nodes for keyless entry are expected to be placed both inside and outside to precisely localize the access device. In the keyless system, these nodes help the car unlock itself and set itself up by adjusting seats, mirrors, powering on the rear-view camera, HVAC, etc., depending on the "profile" of the associated person who is in the vicinity of the car, and also start the car only when the keyfob is ascertained to be inside the car. The 14 UWB nodes are deployed as follows: four exterior bumper nodes (nodes 1, 2, 5 and 6) - i.e. two nodes on either side of the front and rear bumpers, two exterior nodes at the center top on each side (nodes 3, 4), four interior nodes on the four interior corners (nodes 7-10), a node on the rear-view mirror (node 0), a node each behind the glove compartment (node 11), on the interior light switch (i.e. center rooftop) (node 12), and inside the trunk (node 13).

As a first step, in nodes selection, we test connectivity - i.e. we ask the question, *"which nodes can communicate with which other nodes in the car?"* For this, we park the car in an indoor garage, and ask each of the 14 nodes to send 6000 blinks sequentially, and compute the blink delivery rate at each node. We observe that most nodes can communicate with one another except those in the trunk and the front bumper whose connectivity suffers because of the distance and presence of multiple signal attenuators along the path(s). From Figure 6, we observe that the RSS for an interior transmission is on average much larger than a transmission from an interior (or exterior) node to an exterior (or interior) node, (or) between two exterior nodes. This is primarily because of the lack of any significant attenuating objects for an indoor transmission such as metal (car frame). We also observe *link symmetry* - i.e. given a pair of nodes N_i, and N_j, the RSS and blink delivery rates in link $L(N_i,N_j)$ is similar to the one in link $L(N_j,N_i)$.

With these observations, we set the node on the rear-view mirror (node 0) as the transmitting node (tag), and the remaining nodes as receivers (slaves). We choose node 0 as the transmitter for the following reasons : (i) it can communicate with all the nodes in the car at reasonably high power, (ii) it creates a symmetric sensing region in the car from an experimental standpoint (i.e. the observations made for states on one side of the car will translate to the other side of the car), and (iii) every car has a rear-view mirror.

C. CIR Upsampling

As mentioned earlier, each node that receives a blink (transmitted by node 0) will have an associated CIR. Given a Nyquist rate of 1GHz, each CIR tap is 1ns apart. We increase the resolution of this CIR by upsampling [23]. This upsampling process takes us closer to the original analog waveform, and thus helps in more accurate alignment (Section III-D). Figure 7 shows an example of upsampling. Specifically, we take a *fast Fourier transformation* (FFT) of the time-domain CIR y. Next, we zero-pad this frequency domain signal by factor of N*(K-1), where N is the number of taps in the CIR and K is the upsampling factor. Finally, we obtain the upsampled CIR $\hat{y}$ by taking the the inverse-FFT of the upsampled frequency domain signal.

D. CIR Alignment

As the transmit and receive nodes are not synchronized, the CIR frames computed by a receiver node are randomly shifted with respect to one another. Figure 8 shows this misalignment in 5 CIRs computed by a receiver node. While it is possible to mitigate this by synchronizing the nodes via a common reference clock, it will result in added cost and complexity as the keyless setup does not require synchronization for its primary use-case. Consequently, each node performs alignment by identifying an "event" that occurs in all CIRs independent of the environment, and then shifting the location (tap) of that "event" to a reference pivot tap. Said differently, each CIR is shifted differently with the degree of a CIR shift depending on the tap corresponding to the arrival of the first (direct) path. As a result of this shifting, the first path of every CIR computed by a node now occurs at the pivot. We point out that this first path is not necessarily the strongest path and

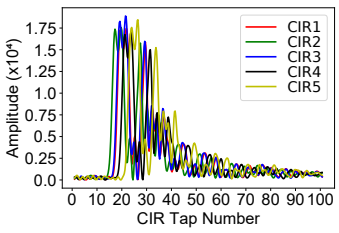

Fig. 8: The CIRs are misaligned due to the absence of transceiver synchronization and receiver hardware imperfections.

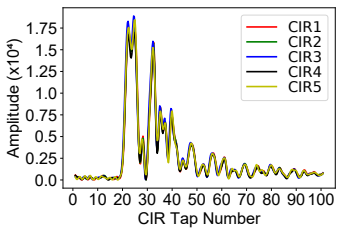

Fig. 9: Each receiver performs CIR alignment by identifying the tap corresponding to the first path, and shifting it to a pivot tap.

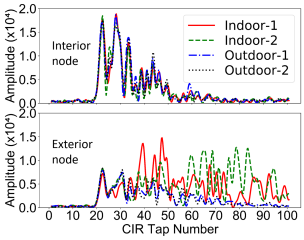

Fig. 10: Raw CIRs for an empty car as observed by an interior and an exterior node at 4 different locations.

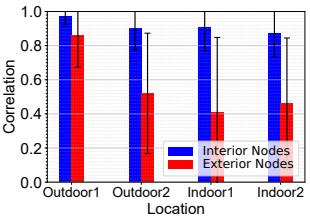

Fig. 11: The interior nodes are more robust to location changes than the exterior nodes.

that the nodes in the bumpers can also observe the first path, albeit heavily attenuated. Figure 9 shows the aligned version of the 5 misaligned CIRs shown in Figure 8. The tap location corresponding to the arrival of the first path, referred to as *First Path Index* (FP_{idx}) is typically exported by UWB chips such as the Decawave DW1000 [26]. This first-path based alignment benefits from upsampling, as the FP_{idx} is at a much finer resolution of 15.625ps [26] (compared to the raw CIR 1ns tap resolution). We also verified the alignment process by computing the "lag" between any two CIRs received by a node via cross correlation. We observed that the "lag" corresponds to the difference between their first path indices.

E. Receiver Selection

Having aligned the CIRs received by a node with respect to each other, we next ask the question, *"which of the remaining nodes can actually become receivers?"* To answer this, we first park the car at 4 different locations - two spots in an indoor garage, an outdoor location with cars on either side and a free-space setup (i.e. the car has nothing in its vicinity). We set node 0 to transmit and the rest to receive. In each location, we collect CIRs of different car states (empty, occupied, door, window and trunk open). Next, we compute the average Pearson correlation coefficient between the CIRs (of a given state) computed in free-space (called *Outdoor1*) and all the 4 locations, for each of the internal and external nodes. This correlation coefficient $\hat{R}$ between two CIRs x and y of duration t taps is given by:

$$\hat{R_{xy}} = \frac{\sum_{i=1}^{t}(x_i - \overline{x})(y_i - \overline{y})}{\sqrt{\sum_{i=1}^{t}(x_i - \overline{x})^2}\sqrt{\sum_{i=1}^{t}(y_i - \overline{y})^2}} \quad (2)$$

where x_i, y_i refer to the CIR amplitude in the i^{th} tap of CIRs x and y respectively, and $\overline{x}$, $\overline{y}$ refer to the sample mean of the two CIRs.

From Figures 10 and 11, we observe that irrespective of the location, the CIR of the internal nodes correlate much higher than their external counterparts. This is because the internal nodes are more robust to location changes than the external nodes. The robustness arises primarily because the metallic car frame acts like a shield from outside reflections. When the doors/windows are open, the exterior reflections potentially incident on some nodes (depending on the angle of reflection) only affect a small portion of the CIR. Consequently, the CIRs of a state continue to correlate better with itself than any other state. Furthermore, we also look only at a narrow CIR window of interest after the arrival of the direct path (Section V-G). Consequently, given the increased robustness to location changes, we use the internal nodes as receivers.

F. State Identification

CaraoKey leverages the CIR to identify the state of the car. At a high level, it uses the fact that the changes in a car state (door open, window open, trunk open, person inside, etc.) alter the multipath reflections inside the car, which is observed in the CIR. For example, an open door will eliminate (or create) reflections that previously existed (or did not exist). Such reflection changes are leveraged to infer the state. While *CaraoKey* is described for a car state sensing application from an intrusion detection standpoint (i.e. when the car is static), it can also be used while the car is in motion as all the transceivers experience the same relative motion. *CaraoKey* performs this car state inference in two steps. In Step 1, each node correlates its observed CIR with reference CIRs (in a maintained corpus) to identify the top-K likely states (Figure 12). Having pruned the state space, *CaraoKey* next extracts features from the CIR – referred to as the *Multipath Profile* (Figure 13), in order to identify the car state. We next explain how *CaraoKey* differentiates the states of interest shown in Figure 14.

1) Step 1: Maximum Likelihood Based State Pruning:: *CaraoKey* leverages the intuition that some states can be captured better by some nodes, while other nodes observe a CIR similar to *Empty*. We capture this intuition in Step 1 via a Maximum Likelihood approach. Here, given an observed CIR (by each node) due to a transmitted blink, each node votes on a particular state based on correlation. *CaraoKey* then computes the likelihood of being in each of the possible states, given this vote. It then fuses the likelihood estimates from all the nodes, to obtain the top-K (we use K=4) most likely states. This is achieved in two phases : a *training phase* and a *testing phase*, as shown in Figure 12.

Training Phase: We explain this phase by first defining some notations. Let $\mathbf{R} = \{\text{R}_1, \text{R}_2, ..., \text{R}_n\}$ be the set of n (receiver) nodes deployed in the car. Let $\mathbf{S} = \{\text{S}_1, \text{S}_2, ..., \text{S}_s\}$ be the set of s car states of interest. Let $\mathbf{C}_i = \{\text{C}_i^{S_1}, \text{C}_i^{S_2}, ..., \text{C}_i^{S_s}\}$ ($1 \leq \text{i} \leq \text{n}$) be a corpus of reference CIRs maintained for each state by a receiver R_i. Each node R_i first builds a likelihood

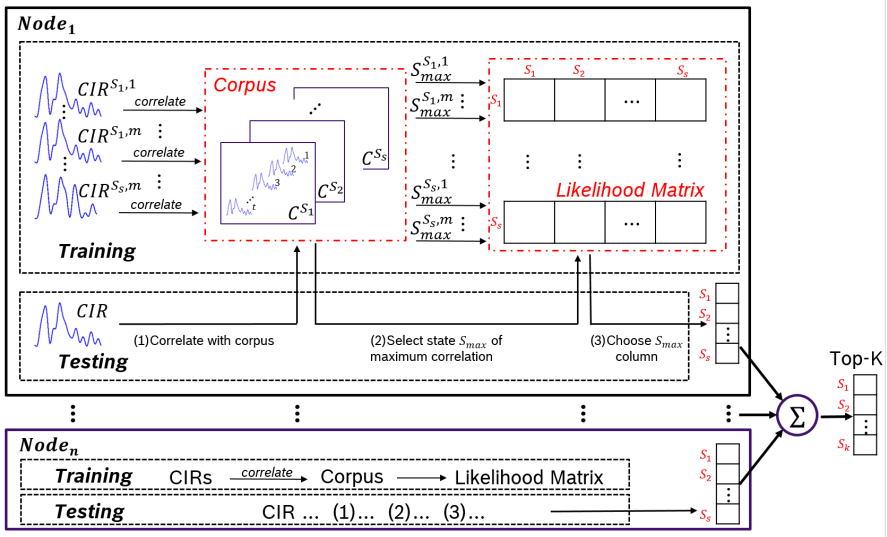

Fig. 12: *CaraoKey* prunes the state space by choosing the Top-K most likely states via a Maximum Likelihood Approach based on CIR correlation with a reference corpus.

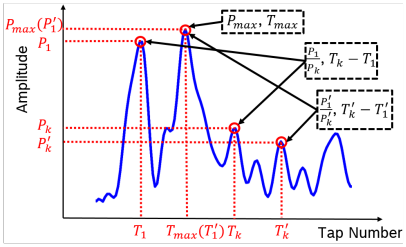

Fig. 13: *CaraoKey* computes a *multipath profile* by extracting peak-based features from each node's observed CIR.

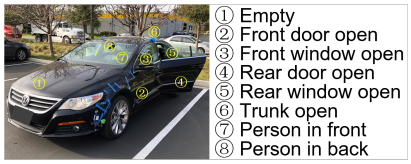

Fig. 14: *CaraoKey* attempts to distinguish 8 car states.

matrix LM_i of dimensions $s \times s$. Each cell (x,y) of the likelihood matrix LM_i essentially denotes the probability of the car being in state S_x, when node R_i votes that the observed CIR is in state S_y (where S_x, $S_y \in \mathbf{S}$). We next explain how a node votes and builds the likelihood matrix.

Given a CIR of state S_x ($S_x \in \mathbf{S}$), a receiver R_i correlates this CIR with (other) CIRs in its corpus $\mathbf{C}_i$, and computes the mean correlation with the reference CIRs of each state. This results in a correlation vector CV_i of dimension s×1. Node R_i then chooses the state of maximum correlation $S^x_{max}(i)$ as its vote. Formally, $S^x_{max}(i)$ = argmax(CV_i). This process is repeated for m different CIRs (m = 50 in *CaraoKey*) of state S_x by the node R_i, resulting in a maximum vote vector $M^x_i = [S^{x,1}_{max}(i), S^{x,2}_{max}(i),...,S^{x,m}_{max}(i)]$. From this vector, node R_i computes a row of the likelihood matrix which can be formally represented as $P(S_x—S_y)$, $\forall\ S_y \in \mathbf{S}$, where $P(S_x—S_y) = \frac{\#\ of\ occurrences\ of\ S_y\ in\ M^x_i}{m}$. Anecdotally, this vector can be understood as: "the probability of being in state S_x (say Empty) when the node votes Empty, node votes front door open, node votes front window open etc. A node that can detect a particular state S_x well will have a high $LM_i(S_x,S_x)$, while a node which cannot detect a particular state well will have $LM_i(S_x,S_x)$ similar to $LM_i(Empty,S_x)$ and $LM_i(S_x,$ Empty). This process is repeated for each of the s states, and for each of the n nodes, resulting in n s×s likelihood matrices.

Testing Phase: In the testing phase, each node R_i correlates its observed CIR with the corpus (as in the training phase), and makes a vote based on maximum correlation ($S_{max}(i)$). From this vote, R_i obtains a likelihood vector (a column of the likelihood matrix), LV_i. Formally, $LV_i = LM_i(S_x, S_{max}(i))\ \forall\ S_x \in \mathbf{S}$. This s-element vector essentially says : 'when node R_i votes $S_{max}(i)$, how likely is the car to be in each of the s states. *CaraoKey* repeats the process for each of the n nodes and then fuses the likelihood vector from each node via a vector sum (i.e. the probability value of being in each state according to every node is summed). *CaraoKey* then passes the top-K most likely states for this observed CIR, on to the next step.

2) Step 2: Multipath Profile based State Inference:: In this step, *CaraoKey* identifies the state of the car. It does so by extracting features from the CIRs observed by the nodes – referred to as the *multipath profile*. We next explain how this *multipath profile* is computed.

As mentioned earlier, the CIR is representative of how the environment impacts the transmitted signal. The peaks in the CIR represent the reflections from the environment. These peaks look different when the state of the car changes. Consequently, *CaraoKey* performs "peak-driven" feature extraction to build the *multipath profile*. These peak-based features are extracted based on position and amplitude. More precisely, as shown in Figure 13, each node extracts the following features from its observed CIR – (i) ratio of the power (amplitude) of the first p peaks - ($\frac{P_1}{P_2}$, $\frac{P_1}{P_3}$, ..., $\frac{P_1}{P_p}$), where P_k refers to the k^{th} peak ordered by position, (ii) ratio of the power of the top p peaks - ($\frac{P'_1}{P'_2}$, $\frac{P'_1}{P'_3}$, ..., $\frac{P'_1}{P'_p}$), where P'_k refers to the k^{th} peak ordered by power and $P'_1 = P_{max}$, (iii) relative (tap) distance between the first p peaks (T_2 - T_1, T_3 - T_1, ..., T_p - T_1), where T_k refers to the tap of the k^{th} peak ordered by location, (iv) relative (tap) distance between the top p peaks (T'_p - T'_1), where T'_k refers to the tap of the k^{th} peak sorted by power such that $T'_1 = T_{max}$, (v) power of the maximum valued peak (P_{max}), (vi) position of the maximum valued peak (T_{max}). (*CaraoKey* uses p = 3). Figure 2 shows an example of a pair of features – $\frac{P_1}{P_2}$ and $\frac{P_{1'}}{P_{2'}}$ from two nodes helping separate many of the states. By adding the remaining features and nodes, *CaraoKey* starts to better distinguish the states of interest. Furthermore, in *Step 1*, the correlation values obtained when the test CIR is correlated with the corpus, are reduced to a single value (max). In this step, the correlation values are also used as features. More precisely, let c^1_i, c^2_i, ..., c^s_i be the mean correlation value obtained by node R_i on correlating the test CIR with elements of the corpus $C^{S_1}_i$, $C^{S_2}_i$,..., $C^{S_s}_i$ respectively.

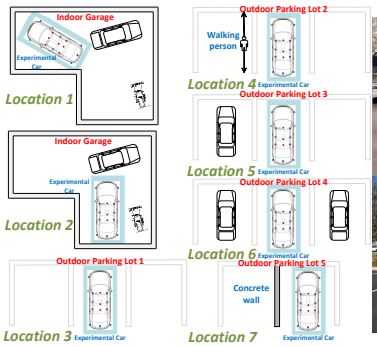

Fig. 15: Experimental locations layout.

Fig. 16: We evaluate *CaraoKey* at 7 different locations (two indoors and five outdoors), under various scenarios.

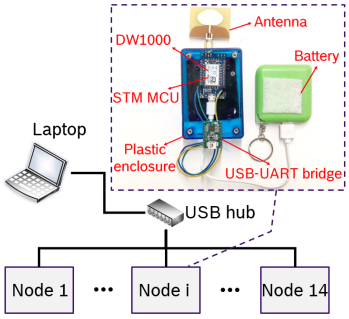

Fig. 17: *CaraoKey* node and data collection system architecture.

As the correlation values from the different nodes are not in the same scale (i.e. correlation values are locally ordered per node, but not globally ordered across nodes), we compute a relative correlation vector : [c_i^1 - c_i^1 , c_i^2 - c_i^1, ..., c_i^s - c_i^1]. Said differently, the relative correlation vector is a measure of change relative to a reference state, namely the *empty state* (i.e. c_i^1 is mean correlation with respect to the corpus of *empty* CIRs).

These features are computed by each of the n nodes, and together referred to as *multipath profile*. The resulting 147-element feature vector is passed through a Random Forest Classifier (100 estimators) which identifies the car state from the K shortlisted states.

IV. Implementation and Experimental Setup

Implementation : Figure 17 shows an overview of our implementation. To implement *CaraoKey*, we program 14 Radino32 Spider boards [27]. Each board consists of a Decawave DW1000 transceiver chip [26] that complies with the IEEE802.15.4 UWB standard [8], and an STM32L151 microcontroller [28] to control the chip. These boards are programmed to output a 250-tap CIR at 33Hz. Each tap is represented as a 4-byte complex number from which the amplitude is computed as its magnitude. We point out that *CaraoKey* only uses 25 of the 250 taps. However, we read out 250 taps to perform sensitivity analysis (Section V-G). These "truncated" CIRs are read from each Radino board via its serial port that is connected to a host laptop via a Silicon Labs CP2102N USB-UART bridge [29] and a USB 3.0 hub. These CIRs are upsampled by a factor of 4 in the host laptop. Each board is terminated by a 3.3dBi omni-directional antenna [30]. Each node is powered via USB through an 1800mAh power bank, and placed inside a plastic enclosure. The boards are programmed to use Channel 5 of the UWB standard - occupying the 6.24 to 6.74 GHz frequency band. We use a 1024-length preamble with a 64MHz pulse repetition frequency. Finally, the CIRs that are read onto the host laptop are processed (Section III) in Python.

Experimental Setup : To test our hypothesis, we mount the 14 nodes as previously shown in Figure 5, on two configurations of a Volkswagen CC. We place the instrumented cars at 7 different locations of varying multipath – two indoors (L1 and L2) (in a multipath rich garage) and five outdoors (L3-L7), as shown in Figure 15 and 16. The five chosen outdoor locations were all different and tested different factors. In L3 the car was parked in free space (i.e. no person or vehicle besides the instrumented car), in L4 a person was walking close (roughly 50cm) to the car, in L5 a car (SUV) was parked on one side, in L6 (different) cars (a hatchback on the left, a sedan on the right) were parked on both sides, and in L7 the car was parked next to a concrete wall. In each location, we collect 3 minutes of data for each state – empty, front door open, front window open, rear door open, rear window open, person in front, person in back and trunk open. The experiments for all states (except empty and trunk open) were performed on the left-side of the car alone. This is because of the symmetric nature of the setup (Section III-B). For the person inside experiments, two different people sat inside the car. During the study, other vehicles and people moved freely in the adjacent lanes of the parking lot (or) adjoining roads. We evaluate *CaraoKey* in terms of two metrics – *accuracy* and *power consumption*. To measure power consumption, we use the Keysight N6705B DC Power Analyzer [31]. From a power dissipation standpoint, there are no health concerns associated with *CaraoKey* as it is UWB-standards compliant. Its Effective Isotropic Radiated Power after accounting for antenna gain and cable loss is just 79μW. In comparison, a typical 5GHz Wi-Fi access point transmits at 200mW which is nearly 2500x higher. (FCC limit is 1W [32]).

V. Evaluation

A. System Accuracy

We first evaluate *CaraoKey's* ability to differentiate the states of interest. In this evaluation, we first perform a leave-one-out cross validation (i.e. train on 6 locations and test on the remaining 1 location), and report the average. From Figure 18, we observe that *CaraoKey* has an average state classification accuracy of about 98% (across all states). Each state individually also has an average classification accuracy of nearly 95% and above. At times, the *Empty* state tends to get confused with one of the window open states (or vice-versa), but not the door open states. This is because doors (unlike windows) are typically made of metal, and hence their open/close actions affect the *multipath profile* more drastically,

	Empty	F-door open	F-win open	R-door open	R-win open	Trunk open	P-in front	P-in back
Empty	**94.72**	0	3.34	0	0	0.01	0	1.93
F-door open	0	**100**	0	0	0	0	0	0
F-win open	0.79	0	**99.21**	0	0	0	0	0
R-door open	0	0	0	**99.71**	0	0	0.02	0.27
R-win open	1.29	0	0.99	0.05	**96.04**	0	0.07	1.56
Trunk open	0	0	0.01	0	0	**99.98**	0	0.01
P-in front	0	0	0	0	0	0	**99.97**	0.03
P-in back	0	0	0	0.01	0	0	0.01	**99.98**

Fig. 18: *CaraoKey* achieves 98% accuracy by performing a leave-one-out cross validation.

	Empty	F-door open	F-win open	R-door open	R-win open	Trunk open	P-in front	P-in back
Empty	**89.27**	0	1.08	0.01	8.03	0.66	0.02	0.93
F-door open	0.03	**97.61**	2.36	0	0	0	0	0
F-win open	8.41	0.12	**90.92**	0.01	0.50	0.04	0	0
R-door open	0	0	0	**99.28**	0.21	0	0	0.51
R-win open	5.55	0	0.49	0.63	**91.95**	1.35	0.01	0.02
Trunk open	2.19	0	0.05	0	1.24	**96.52**	0	0
P-in front	0.04	5.57	1.11	0.54	0.09	0.35	**91.41**	0.89
P-in back	0.13	0.32	0.08	13.90	0.56	0.08	1.94	**82.99**

Fig. 19: *CaraoKey* achieves over 92% accuracy by training on just 1 location.

thus making them more distinct from the *empty* state. Figure 19 shows the individual state classification accuracy under pure location agnosticism. Here the location under test uses training data from just one location. Each cell in the matrix is the average state classification accuracy of all 42 possible combinations per dataset ($^7C_1 \times 6$) of training on 1 location, and testing on one other location. Even under such conditions, *CaraoKey* achieves an average state classification accuracy of 92% (across all states). This accuracy is lower than the leave-one-out scenario because of the absence of observation of similar scenarios in training. *CaraoKey* increases the accuracy by adding more diverse training data. However, the accuracy can also be improved by increasing the number of transmitters (i.e. a round-robin set of transmitters). If *N* is the number of nodes in the car, then this creates N(N-1)/2 links that can be sensed, instead of (N-1), albeit at the cost of complexity. We leave it as a future work to increase the number of transmitters. Finally, we point out that there is a 7% increase in accuracy by using the 2-step state identification approach (as opposed to just one of the steps).

B. Baseline Comparison

We compare *CaraoKey's* state classification performance with two baselines – an RSS-based baseline and a *CaraoKey* variant that uses only the exterior nodes. In this comparison, all systems employ leave-one-out cross validation.

RSS-based baseline: This baseline is representative of keyless systems that can only use received signal strength (RSS) information - for e.g., keyless systems built on technologies like Bluetooth whose standards currently only export RSS [33]. We emulate such a system by computing the RSS directly from the CIR (Section 4.7.2 of [26]). Using RSS alone, the average accuracy drops down to nearly 49%. While *CaraoKey* benefits from using features derived from the CIR which provide information about the multipath within the car, the RSS is a single aggregate metric with no information about individual paths incident on a receiver. Furthermore, the RSS also does not change significantly between states (roughly 3dBm). As seen in Figure 20, even the top-5 classification accuracy (the fraction of blinks where the ground truth state is in the top-5 most likely states of the classifier) of an RSS-based system is lower than *CaraoKey's* top-1 accuracy.

***CaraoKey* with exterior nodes :** As *CaraoKey* only uses internal nodes (Section III-E), we compare its results with a *CaraoKey* variant that only uses the external nodes. As seen in Figure 20, using the exterior nodes causes the average accuracy to drop to 70%. This is because as mentioned earlier the exterior nodes are less robust to location changes than the interior nodes. Hence, at two different locations, a given state S_i looks "more similar" to another state S_j, than itself. However, the external node variant still has a higher (top-1) accuracy than the RSS-baseline as it uses the CIR which contains more details than the aggregate RSS metric.

C. Effect of Number of Training Locations

We evaluate how *CaraoKey's* accuracy changes as we vary the number of training locations. If *t* is the number of training locations, we generate all $^7C_t \times (7\text{-}t)$ scenarios of training and testing, and compute the classification accuracy across all states, for each scenario, per dataset. From Figure 21, we observe that even if we train on just one location (i.e. pure location independence), we can obtain an average state classification accuracy of 92%. We also observe that as we start to increase the number of diverse training locations, the accuracy starts to increase, with over 95% accuracy by training on any two locations. With 50 training CIRs per state (Section III-F2), 14 states (states of interest can happen on either side of the car) and 6 training locations, the total training time for *CaraoKey* is only 127s ($=\frac{50}{33} \times 14 \times 6$). We point out that such a training can be performed on only one model of a car (in the factory), and not necessarily on every shipped model of a car.

D. Effect of Number of Internal Nodes

We next study how the number of internal receiver nodes affects the accuracy of *CaraoKey*. This attempts to answer the question *"given a node budget by a car manufacturer, what is the highest achievable state classification accuracy?"* For this evaluation, we vary the number of nodes and perform a leave-one-out cross validation. Furthermore, for a given node count (N) we take the (a)symmetry properties of our experiments into account. For example, as we experiment with only the left front door open, and in the scenario where we consider only one front node, we emulate the effect of either door being open by taking the average accuracy of both the front nodes. Hence, say for N = 1, the highest average state classification accuracy (ASCA) is given by max (ASCA(Node 7,Node 8), ASCA(Node 9, Node 10), ASCA(Node11), ASCA(Node12), ASCA(Node13)). From Figure 22, we notice that in general the ASCA increases as we add more nodes. We also observe that with just 3 internal nodes, *CaraoKey* can achieve nearly 94% accuracy in the best arrangement of the 3 nodes. Figure 22 also shows the node combination which gives us the highest ASCA for a given node count. We observe that the trunk node is the most important receiver node, if only one receiver is permitted. This is because it is central and can receive reflections from both sides of the car, and the trunk itself. Similar to Section V-A, we point out that the accuracy for a given receiver node count can potentially be improved by increasing the number of transmitters, thus creating more links for sensing.

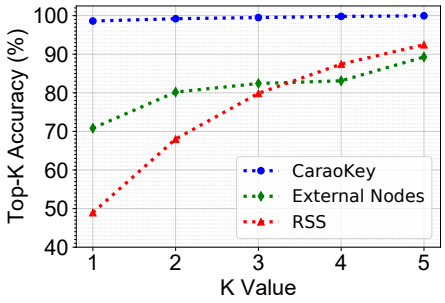

Fig. 20: *CaraoKey* outperforms baselines that use RSS and the exterior nodes by 49% and 28% respectively.

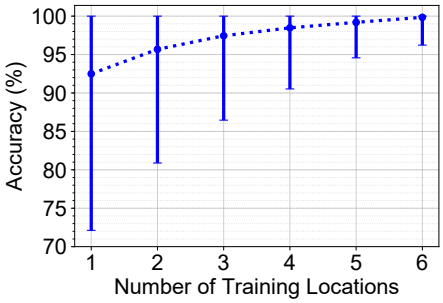

Fig. 21: As the number of training locations increase, the accuracy increases. It achieves 92% accuracy with just 1 training location.

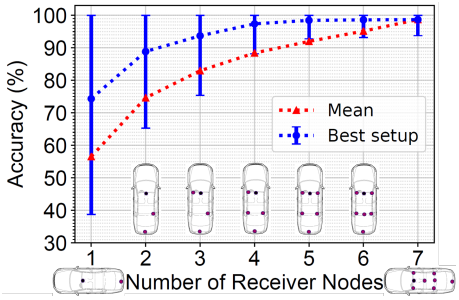

Fig. 22: As the number of receivers increase, the accuracy increases. With just 3 receivers, *CaraoKey* can achieve nearly 94% accuracy.

E. Power Analysis

Based on our measurement via the Keysight N6705B DC Power Analyzer [31], the average current consumption of a transmitter and receiver during a transmit and receive event is 80.5mA and 94mA respectively. By duty-cycling the device (sleep current draw is 2μA) between the blinks, the average current consumption of the transmitter and receiver reduces to 12.6mA and 57.8mA respectively. From Section V-D, we observed that we can obtain 94% accuracy using just 3 receivers (4 nodes in total). This results in a current draw of 186mA (= 12.6 + 57.8*3) at 33Hz blink rate. A typical car battery is expected to run for about 30 days without recharging (when car is parked for extended duration). This results in dark current requirements for the car depending on its make/model and car battery size (typically 50mA with 12V 50Ah battery [34]). Dark currents include all leakage currents and average current consumption of other car sub-systems which are duty cycled when car is 'off', examples include keyless and security sub-systems. To ensure practicality of the solution, it's important that average current load of *CaraoKey* when car is off should not exceed allotted budget (5-10mA). *CaraoKey* can make prediction based on each blink independently. It can reduce current consumption by lowering the blink rate. This is called *down-sampling in slow time* [35]. The power and current consumption with lower blink rates are shown in Figure 23. For a 3 receivers (4 nodes) setup, up to 2Hz blink rate is possible.

F. Effect of Tap Resolution

CaraoKey uses the DW1000 chipset which has a 1ns tap resolution. We study *CaraoKey's* working as we lower this tap resolution. This is called *fast time down-sampling* [35]. This can potentially further reduce power consumption as it places a lower stress on the ADC. We simulate a fast-time down sampling rate of N, by taking every N^{th} CIR tap. From Figure 24, we observe nearly 90% accuracy after downsampling by a factor of 2 (= increasing tap resolution to 2ns). However after that, we notice that the accuracy decreases rapidly as we downsample in fast time. This is because multiple peaks start to fuse as one (or are missed), and the multipath profile that is leveraged to distinguish states becomes no longer distinguishable.

G. Effect of CIR Window Length

CaraoKey looks at a window of 100 taps, (roughly 6m two-way distance after upsampling) to compute the multipath profile. We next study the effect of tap duration, as a smaller window places lesser stress on execution. We observe that as we start to shrink this tap window, the accuracy starts to decrease. This is because with a smaller window there are lesser peaks to build the multipath profile and the profile for the different states start to look similar to one another. Similarly as we start to increase the window length the accuracy begins to decrease. This is because the "noisy" part in the tail of the CIR begins to dominate, and the CIRs (of different states) start to correalte with one another.

VI. Discussion

Effect of Configuration Changes: The *multipath profile* can be affected by configuration changes - i.e. changes in the positions of the seats, mirrors, etc. However, this is typically not an issue, as most modern cars have a notion of "memory" [36]. Consequently, whenever the car is locked, the car can move its seats and mirrors back to its preset configuration.

Effect of Objects: We evaluate *CaraoKey's* state identification in the presence of objects inside the car. With small commonly used objects like backpacks, the accuracy is not impacted. With a large box placed inside the car, *CaraoKey* mis-classified the *empty* car to have a person inside 40% of the time. This is addressed by fusing data from multiple CIRs and measuring the variance of a given tap (across time).

Use of Dedicated Sensors: Depending on the make and model, some car states can be determined via the use of dedicated sensors too. However, *CaraoKey* still needs to "understand" these states because the keyless infrastructure is independent from other sensor systems. This "self-compiled" knowledge of car states can be used as the building block for other sensing applications based on keyless infrastructure.

Effect of multi-state and sub-states: In the current version, we can only detect a single state. This suffices for many practical scenarios involving a parked car, where unexpected states will happen in succession and not altogether. However, a car can potentially be in more than one state or can have partially open doors, windows too. This can be handled in two

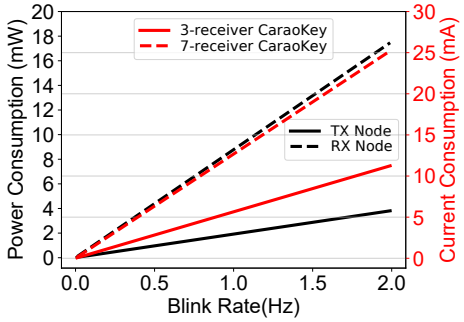

Fig. 23: Power performance of *CaraoKey* with blink rate up to 2Hz.

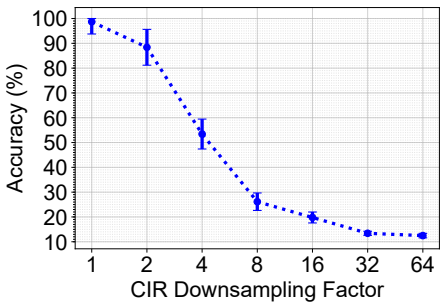

Fig. 24: As a CIR is downsampled in fast time, the accuracy starts to decrease.

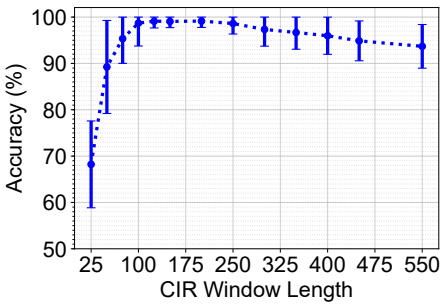

Fig. 25: *CaraoKey* performance with different CIR window lengths.

ways - (i) training for combination of states or adding sub-states, or (ii) using the probability values of the classification process to identify the multi-state or sub-states possibility.

VII. Conclusion

With car manufacturers developing UWB-based solutions for keyless entry, this paper is the first to explore the possibility of using this UWB keyless infrastructure for secondary use-cases. More precisely, in this paper, we present *CaraoKey*, a sensing system that estimates the car state via the UWB keyless infrastructure. It does so by leveraging the multipath information contained in the CIR computed by an UWB receiver. We implement a 14-node setup and evaluate it in 7 different locations and scenarios. Our results indicate that *CaraoKey* can detect the car state with 98% accuracy using 8 nodes, and 94% accuracy using just 4 nodes. *CaraoKey* is an example of using the UWB keyless infrastructure as a sensing modality. This infrastructure can also be used to build several other applications like monitoring the vital signs of the occupants, performing occupancy counting, in-car activity recognition, etc. Such features will become increasingly useful in future driverless cars, shuttles and taxis where passengers' in-vehicle security and well-being will become increasingly important.

References

[1] T. J. Waraksa et al., "Passive keyless entry system," Jul. 17 1990, US Patent 4,942,393.

[2] A. Francillon et al., "Relay attacks on passive keyless entry and start systems in modern cars," in *NDSS Symposium*, 2011.

[3] A. Humayed et al., "Cyber-physical security for smart cars: taxonomy of vulnerabilities, threats, and attacks," in *ACM ICCPS*, 2015.

[4] "Tesla stolen via keyfob hack," https://electrek.co/2018/10/21/tesla-stealing-video-keyfob-hack/, Mar. 2020.

[5] "Automotive news: NXP and VW share the wide possibilities of UWB fine ranging capabilities," https://media.nxp.com/news-releases/news-release-details/nxp-and-vw-share-wide-possibilities-ultra-widebands-uwb-fine, Mar. 2020.

[6] "Automotive security: Why UWB measures up," https://www.decawave.com/automotive-security-why-uwb-measures-up-eecatalogue-us/, Mar. 2020.

[7] "FIRA consortium : UWB use cases," https://www.firaconsortium.org/discover/use-cases, Mar. 2020.

[8] "IEEE standard for low-rate wireless networks," *IEEE Std 802.15.4-2015 (Revision of IEEE Std 802.15.4-2011)*, pp. 1–709, Oct 2016.

[9] "IEEE 802.15 WPAN task group 4z enhanced impulse radio," http://www.ieee802.org/15/pub/TG4z.html, Mar. 2020.

[10] M. Kotaru et al., "Spotfi: Decimeter level localization using wifi," in *ACM SIGCOMM*, 2015.

[11] Y. Zeng et al., "Wiwho: Wifi-based person identification in smart spaces," in *ACM/IEEE IPSN*, 2016.

[12] R. Nandakumar et al., "Fingerio: Using active sonar for fine-grained finger tracking," in *ACM CHI*, 2016.

[13] T. Li et al., "Practical human sensing in the light," in *ACM MobiSys*, 2016.

[14] K. Golsch et al., "Passive entry/passive start systems and methods for vehicles," Jun. 25 2019, US Patent 10,328,898.

[15] A. E. Assaad, "Angle detection for locating mobile terminals," Aug. 3 2018, DE 102017214435.1.

[16] R. Mazraani et al., "Experimental results of a combined TDOA/TOF technique for UWB based localization systems," in *IEEE ICC Workshops*, 2017.

[17] Y. Zeng et al., "Your AP knows how you move: Fine-grained device motion recognition through WiFi," in *ACM HotWireless*, 2014.

[18] Y. Chen et al., "FM-based indoor localization," in *ACM MobiSys*, 2012.

[19] P. Melgarejo et al., "Leveraging directional antenna capabilities for fine-grained gesture recognition," in *ACM UbiComp*, 2014.

[20] J. Yang et al., "Detecting driver phone use leveraging car speakers," in *ACM MobiCom*, 2011.

[21] X. Xie et al., "Wireless CSI-based head tracking in the driver seat," in *ACM CoNEXT*, 2018.

[22] M. Raja et al., "Detecting driver's distracted behaviour from Wi-Fi," in *IEEE VTC Spring*, 2018.

[23] A. Dhekne et al., "LiquID: A wireless liquid IDentifier," in *ACM MobiSys*, 2018.

[24] H. Mohammadmoradi et al., "Room occupancy estimation through WiFi, UWB, and light sensors mounted on doorways," in *ACM ICSDE*, 2017.

[25] V. Ipatov, "Ternary sequences with ideal periodic autocorrelation properties," *Radio Engineering and Electronic Physics*, vol. 24, 1979.

[26] "Decawave DW1000 user manual," https://www.decawave.com/sites/default/files/resources/dw1000_user_manual_2.11.pdf, Mar. 2020.

[27] "In-Circuit Radino spider datasheet," https://wiki.in-circuit.de/images/8/85/305000086A_radino_Spider_RP-SMA.pdf, Mar. 2020.

[28] "STM32L151 datasheet," https://www.st.com/resource/en/datasheet/cd00277537.pdf, Mar. 2020.

[29] "Silicon labs CP2102n USB-to-UART bridge," https://www.silabs.com/documents/public/data-sheets/cp2102n-datasheet.pdf, Mar. 2020.

[30] "In-Circuit ultra-wideband (UWB) antenna, 3-8GHz," https://shop.in-circuit.de/product_info.php?cPath=22_26&products_id=187, Mar. 2020.

[31] "Keysight N6705B DC power analyzer," https://www.keysight.com/en/pd-1842303-pn-N6705B/dc-power-analyzer-modular-600-w-4-slots, Mar. 2020.

[32] "Electronic code of federal regulations," https://bit.ly/2MBPgF8, Mar. 2020.

[33] "Bluetooth 5.0 standard," https://www.bluetooth.org/DocMan/handlers/DownloadDoc.ashx?doc_id=421043, Oct. 2019.

[34] "What is 'normal' for a parasitic draw?" https://www.optimabatteries.com/en-us/experience/2010/12/what-normal-parasitic-draw, Jan. 2020.

[35] M. A. Richards, *Fundamentals of radar signal processing*. Tata McGraw-Hill Education, 2005.

[36] "2018 Acura NSX personalized settings," http://m.acura.com/pdf/owners/2018/NSX/2018_NSX_Personalized_Settings.pdf, Oct. 2019.