Non-Cooperative Wi-Fi Localization & its Privacy Implications

Ali Abedi
University of Waterloo
Canada
ali.abedi@uwaterloo.ca

Deepak Vasisht
University of Illinois Urbana-Champaign
United States of America
deepakv@illinois.edu

ABSTRACT

We present Wi-Peep – a new location-revealing privacy attack on non-cooperative Wi-Fi devices. Wi-Peep exploits loopholes in the 802.11 protocol to elicit responses from Wi-Fi devices on a network that we do not have access to. It then uses a novel time-of-flight measurement scheme to locate these devices. Wi-Peep works without any hardware or software modifications on target devices and without requiring access to the physical space that they are deployed in. Therefore, a pedestrian or a drone that carries a Wi-Peep device can estimate the location of every Wi-Fi device in a building. Our Wi-Peep design costs $20 and weighs less than 10 g. We deploy it on a lightweight drone and show that a drone flying over a house can estimate the location of Wi-Fi devices across multiple floors to meter-level accuracy. Finally, we investigate different mitigation techniques to secure future Wi-Fi devices against such attacks.

ACM Reference Format:
Ali Abedi and Deepak Vasisht. 2022. Non-Cooperative Wi-Fi Localization & its Privacy Implications. In *The 28th Annual International Conference On Mobile Computing And Networking (ACM MobiCom '22), October 17–21, 2022, Sydney, NSW, Australia.* ACM, New York, NY, USA, 13 pages. https://doi.org/10.1145/3495243.3560530

1 INTRODUCTION

We live in an era of Wi-Fi connected TVs, refrigerators, security cameras, and smart sensors. We carry personal devices like smartwatches, smartphones, tables, and laptops. Due to the deep penetration of Wi-Fi devices in our lives, location privacy of these devices is an important and challenging objective. Imagine a drone that flies over your home and detects the location of all of your Wi-Fi devices. It could infer the location of home occupants, security cameras and even home intrusion sensors. A burglar could use this information to locate valuable items like laptops and identify ideal opportunities when people are either not at home or away from a specific area (e.g., everyone is in the basement) by tracking their smartphones or smartwatches. The promise of pervasive connectivity has been to merge our physical and digital worlds, but the leakage of such location information brings arguably the worst aspect of the digital world – pervasive tracking – to the physical world.

In this paper, we show that there are fundamental aspects of the Wi-Fi (IEEE 802.11) protocol that leak such location information to a potential attacker. We demonstrate that it is possible to reveal accurate location of all Wi-Fi devices in an indoor environment (a) non-cooperatively – without any coordination with Wi-Fi devices or the access points, (b) instantaneously – without waiting for devices to organically transmit packets, and (c) surreptitiously – without any complex infrastructure deployment in the surrounding. Our goal is to expose the security and privacy vulnerabilities of the 802.11 Wi-Fi protocol by demonstrating a first-of-its-kind non-cooperative localization capability. We hope that our work will inform the design of next-generation protocols.

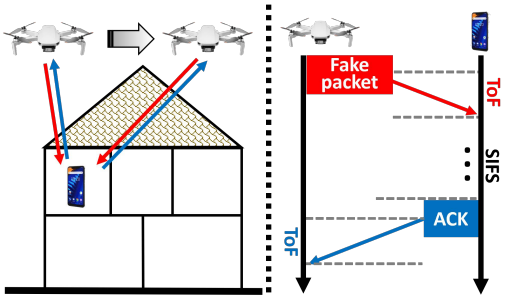


Figure 1: Overview of Wi-Peep

We note that there has been much past work in Wi-Fi-based positioning. However, such past work does not enable non-cooperative, surreptitious localization of Wi-Fi devices. First, most of this work [5, 22, 25, 36, 42] relies on co-operation from end devices – e.g., the client needs to switch channels [36] or move [25] or share inertial sensor data [25]. Second, state-of-the-art techniques, such as ArrayTrack [39], rely on antenna arrays with multiple antennas, that are typically bulky and cannot be easily carried by a person or a small drone. Deploying multiple such antenna arrays near a target building makes the attack less practical and easier to detect. Third, RSSI-based techniques [5, 42] rely on fingerprinting or trained models that require physical access to the target space. Finally, most of these need client devices to continuously transmit Wi-Fi packets or share their received Wi-Fi packets by installing an application, an access we cannot assume for such privacy revealing mechanisms.

We present, Wi-Peep, a system that is quick, accurate, and performs non-cooperative localization. It does not require any access to target devices or the network access point. It does not even need the attacker to connect to the same Wi-Fi network. In our attack, shown in Fig. 1, the attacker (e.g., a light-weight drone or a pedestrian) passes by the house carrying a small Wi-Fi capable device and estimates the location of all Wi-Fi devices in the target environment. We exploit the design of the 802.11 protocol to first generate Wi-Fi traffic from non-cooperative clients and then use a novel time-of-flight based technique to locate these devices. Wi-Peep solves the following challenges:

ACM MobiCom '22, October 17–21, 2022, Sydney, NSW, Australia

ACM ISBN 978-1-4503-9181-8/22/10…$15.00
https://doi.org/10.1145/3495243.3560530

(i) Generate Wi-Fi traffic without cooperation – We must (a) identify all devices in the network quickly at the start of the attack, and (b) generate Wi-Fi traffic continuously from such devices to perform location estimation. A simple solution to identifying devices is to passively wait for Wi-Fi devices to transmit a packet. This approach is problematic because it requires the attacker to linger around for a long time. Instead, we exploit the 802.11 power saving mechanism (available in all 802.11 standards from 11a/b to 11ax) by injecting a fake beacon imitating the access point that tells all connected Wi-Fi devices to contact the access point for buffered packets. This beacon elicits a response from all devices in the target Wi-Fi network, as described in Sec. 4.1. Once we have identified all devices, we use targeted packets to each of these devices. To perform time of flight measurements on these devices, the attacker requires exchanging packets directly with target devices. Therefore, natural traffic from a target device cannot be used. Recent work [1] has shown that 802.11 devices always respond to packets with an ACK, even when the packets emerge outside the Wi-Fi network and are unencrypted/incorrectly encrypted. We use this flaw to perform ToF measurements to any target device. The challenge in using Polite Wi-Fi is that Wi-Fi devices are in the sleep mode most of the time and their radio is turned off. We have designed a technique that allows an attacker to keep the radio of target devices on during the attack so that they keep sending ACKs.

(ii) Localization despite noisy SIFS – In 802.11, ACKs are sent at a fixed interval after receiving a data packet. This interval is called Short Interframe Space or SIFS as illustrated in Figure 1. Wi-Peep measures the round trip time between a packet transmission and ACK reception and subtracts the SIFS. This allows Wi-Peep to estimate the time-of-flight and hence the distance between the attacker and a target device. Unfortunately, our experiments reveal that even though the Wi-Fi protocol mandates SIFS to be 10 μs, in practice, this delay can vary from 8 to 13 μs. Such errors can randomize the location estimation process. We build a new algorithm to correct for such variations in time-of-flight estimates (Sec. 4.3).

(iii) How to deal with the multipath effect –The ToF measurements are error-prone because multiple copies of a signal arrive at the receiver from multiple paths. The strongest path may not necessarily be the direct path. Since the attacker is far away and obstructed from the target, this problem is further exacerbated. Indeed, our measurements reveal that Wi-Peep's individual time-of-flight measurements are error prone due to this reason. To counter this challenge, we take the 'wisdom-of-the-crowd' approach. Even though each measurement is noisy, Wi-Peep involves quick packet-ACK sequences at the millisecond level. Therefore, we can collect hundreds of measurements as the attacker flies by (or walks by) the target building. We exploit the spatial diversity of these measurements to get an accurate position estimate of our targets.

We have implemented our design on an ultra-light DJI mini 2 drone [7] using off-the-shelf ESP32 [9] and ESP8266 [10] Wi-Fi modules. Our hardware weighs 10 grams, and costs sub-20 dollars. It can be deployed on lightweight drones or carried by a person. Our evaluations in a real environment shows that Wi-Peep finds the location of target devices in an 802.11ax Wi-Fi 6 network on 3 different floors of a house with a median error of 1.2 meters in around two minutes. The contributions of this paper are:

- We present a new way for using 802.11 protocol features to perform time-of-flight based positioning of Wi-Fi devices without having any control over target devices.
- We find that many devices deviate from the standard time for SIFS which creates a challenge for localization. We design a localization technique that finds a target device without knowing the exact SIFS used by the device.
- We present a solution for future WiFi chipsets that allows authenticated devices to perform localization, while disabling non-cooperative attacks.

2 WI-PEEP SCOPE AND IMPLICATIONS

The Wi-Peep attack works with any Wi-Fi device without instrumentation, i.e., without any application or firmware-level changes. It does not need physical access to the enclosed physical space and does not need to break the encryption of the Wi-Fi network. Once the target MAC address is obtained, the target device doesn't even need to be connected to Wi-Fi. Due to the ease of attack, Wi-Peep has many privacy and security implications. We list some example implications below. In these scenarios, we assume that it is common for a person to carry a Wi-Fi capable device such as a smartphone or a smartwatch. Also, note that the type of a device (e.g. iPhone vs smart sensors) can be identified through various means like the vendor specific information in the MAC address.

- [Security] An attacker can track the location of security guards inside sensitive building (e.g. banks) if they carry a smartphone or a smartwatch.
- [Privacy] An eavesdropper can fly a drone over a hotel to find the number and types of rooms currently occupied. This can be done by a rival hotel trying to find detailed information of how the target business is performing. WiFi devices that belong to a room such as smart TVs can be filtered based on MAC addresses. If other devices such as tablets and laptops are found in a room, it can be considered occupied. This can be done in the middle of the night when most guests are in their rooms.
- [Privacy/Security] If the MAC address of a device that belongs to a person of interest is known, Wi-Peep can track that person in a crowd or inside a building like a shopping center or an airport (even when their device is not connected to any Wi-Fi network).
- [Security] Wi-Peep could be used by burglars to find out the occupancy status of specific parts in a building. For example, the burglar can find out that all people are on the second floor and the basement is empty.

Wi-Peep can also be used for positive use-cases. For example, in a hostage situation, the police can fly a drone over the building to find out where the hostages are kept because many hostages might have smart devices on them. It might also be possible to track the attacker(s) as well.

3 RELATED WORK

Location leakage of wireless devices has been studied from multiple perspectives in the past [16, 19, 32–34]. For example, Wi-Fi devices periodically broadcast probe requests to check for nearby available access points [11]. This reveals the presence of the Wi-Fi device within the range of a sniffer. To avoid this leakage, multiple chipset vendors implemented MAC address randomization [35]. In

contrast to past work, our design is unique in its ability to offer sub-room-level localization accuracy, without requiring access to the end device and the ability to operate in an outside-in manner (the eavesdropper can be outside the home). MAC randomization does not stop Wi-Peep because the device-discovering mechanism finds the randomized MAC address of target devices. The randomized MAC address does not change after association.

Our work builds on past work in indoor localization. Indoor localization methods can be broadly divided into two categories – fingerprinting/training-based methods [5, 15, 27, 28, 42] and training-free methods (e.g. Angle-of-arrival [13, 21, 22, 37–39], time-of-flight [25, 36, 40, 41]). Training-based methods require collection of extensive training data in an environment and therefore, are inapplicable for privacy attacks such as Wi-Peep.

Angle-of-arrival methods [39] typically require multiple antenna arrays. Antenna arrays are generally hard to deploy inconspicuously since they need to span multiple wavelengths (span 25 to 50 cm for Wi-Fi) to enable accurate angle computation. Some past work [22] leverages motion to create virtual antenna arrays. We note that this motion needs a large span with one end fixed (to remove carrier frequency offsets), and therefore has similar issues in terms of weight and size. In contrast, our design is lightweight, does not require multiple antennas or large size. More fundamentally, AoA based localization suffers from large errors when measured from a distance because a small error in angle translates to a large error in location. Since we aim to identify location from a distance outside the home, AoA-based methods are prone to large errors.

The closest line of work to our approach are ranging-based methods that directly compute distance, e.g., Chronos [36] and SAIL [25]. Chronos requires the client and access point to switch Wi-Fi channels in a coordinated fashion and share measured channel information, both of which are not possible in privacy attacks. SAIL, on the other hand, relies on time delay between a transmitted packet and its acknowledgment computed using the channel impulse response (CIR). It builds on this primitive and combines information from inertial sensors on the target phone (which we don't have access to). In addition, SAIL does not account for variation in per-device SIFS, which is a key contribution of Wi-Peep.

Recently, the 802.11 standard has proposed FTM (Fine time measurements) [14, 17, 18] to enable localization of smart devices. In principle, FTM works on the same intuition as Wi-Peep. It timestamps packet departure, arrival at target, ACK departure from target, and ACK arrival at the sender. By combining these four time-stamps, it outputs a time-of-flight estimate. First, note that FTM does not have to deal with variable SIFS delays because it can timestamp packets at both ends (sender and receiver), and therefore needs coordination from the target device (unlike Wi-Peep). In addition, FTM is not widely available yet. Finally, FTM would suffer from the same multipath-like errors that Wi-Peep deals with. Our techniques to deal with such errors are generalizable to FTM and other such ranging based methods.

Finally, we note that there has been recent work in sniffing location of users inside homes using passive and active sensing. For example, [43] uses a Wi-Fi sensor deployed outside a home to infer occupancy of individual rooms. Our work derives inspiration from that work but is orthogonal in its approach. We do not need a Wi-Fi sensor deployed for long durations – our sensor can just fly by on a drone. Wi-Peep does not require frequent organic transmissions from in-home devices. Finally, Wi-Peep deals with new technical challenges like multipath effects and SIFS variability.

4 SYSTEM DESIGN

Wi-Peep operates in three steps. First, we discover the MAC address of target devices. In Section 4.1, we explain how a Wi-Peep eavesdropper can exploit the 802.11 power saving mechanism to discover all Wi-Fi devices connected to an access point in a few seconds. Next, the eavesdropper performs time of flight measurements to one or multiple target devices using the procedure in Section 4.2. Finally, Wi-Peep utilizes time of flight measurements to localize target devices. In Section 4.3 we present a model for accurate localization of target devices despite the noisy ToF measurements and the unknown value of SIFS each device uses.

4.1 Identifying Target Devices

To identify target devices and their MAC addresses, the Wi-Peep attacker first identifies the SSID of the target Wi-Fi network. This can be obtained by a few different methods. In some cases, the target network might have a specific name that reveals which home it belongs to. If this method is not applicable, the attacker can find the SSID of the target house by getting close to the target house and finding the Wi-Fi network with the strongest signal strength. Note, Wi-Peep does not need the password of the target network.

After obtaining the SSID of the target network, the attacker needs to find the MAC address of the devices in this network. Although the attacker can passively sniff packets in this network to discover its Wi-Fi devices, it may not be an effective technique in practice. This is because some devices do not transmit any packet for long periods of time. As a result, the attacker has to wait for a long time to discover all devices. To quantify this wait time, we monitor the natural traffic of a home Wi-Fi network that consists of a variety of Wi-Fi devices including, smartphones, tablets and security cameras for 12 hours. We notice that although most devices transmit a packet every 10 to 20 seconds, some devices may not send any packet for a long time. For example, in our experiments a Samsung S7 phone and Microsoft Surface tablet sometimes do not send any packet for more than 10 minutes. As a result, if an attacker passively sniffs packets for a short while, some devices will not be discovered. We will later show in this section that our device discovery technique can force these two devices to transmit a packet in just a few seconds. Note that Wi-Peep requires only a few minutes of data collection and even a few minutes of waiting to discover Wi-Fi devices significantly increases the attack duration.

To discover all devices in a Wi-Fi network quickly, Wi-Peep exploits the 802.11 power saving mechanism. The 802.11 standard allows Wi-Fi devices to turn off their radios and go to sleep periodically to save power. When a device is in the sleep mode, the access point buffers all incoming packets for that station and notifies it in the next beacon frame[1]. There is a bitmap in every beacon frame called Traffic Indication Map (TIM) which indicates which devices have buffered packets at the access point. When a Wi-Fi device receives a beacon frame that indicates there are buffered packets

[1]To maximize power saving Wi-Fi devices are allowed to skip a few beacons frames. However, they eventually wake up to check for any incoming packets.

#	MAC Address
1	c4:9d:ed:13:e5
2	dc:ef:ca:45:c2:
3	dc:72:9b:e9:9d:
4	58:d5:0a:6b:fe:
5	d4:e6:b7:54:ba:
6	2c:0e:3d:ba:21:
7	94:e4:ba:4a:75:

(a) Ground truth

Fake beacon

Discovered devices

Source	Destination	TIM	Info
4c:9e:ff:9f:02:	ff:ff:ff:ff:ff:ff	ff	Beacon frame,
dc:ef:ca:45:c2:	4c:9e:ff:9f:02:		Null function
58:d5:0a:6b:fe:	4c:9e:ff:9f:02:		Null function
2c:0e:3d:ba:21:	4c:9e:ff:9f:02:		Null function
94:e4:ba:4a:75:	4c:9e:ff:9f:02:		Null function
dc:72:9b:e9:9d:	4c:9e:ff:9f:02:		QoS Null func
c4:9d:ed:13:e5:	4c:9e:ff:9f:02:		QoS Null func
d4:e6:b7:54:ba:	4c:9e:ff:9f:02:		Null function

(b) Discovered Wi-Fi devices

Figure 2: a) List of Wi-Fi devices in the target network (not available to the attacker). (b) Wi-Fi traffic captured by Wireshark. The attacker sends fake beacon frames with TIM set to "FF" forcing all devices to respond.

for that station, it sends a packet to the access point to notify that the device is now ready to receive the buffered packets.

An attacker running Wi-Peep takes advantage of this power saving mechanism to discover all Wi-Fi devices in a network. We forge and inject a beacon frame that seems to be coming from the target network's access point. The attacker sets the TIM bitmap to FF[2] which means all devices believe they have buffered packets at the access point. As a result, all Wi-Fi devices in the target network respond by sending a packet to the access point notifying it that they are ready to receive. The attacker sniffs these packets to discover all connected devices. Note that the power saving mechanism is implemented in *all* 802.11 a/b/g/n/ac/ax standards. Therefore, all existing devices are vulnerable to this attack.

To illustrate the effectiveness of this technique, we perform this attack on a Wi-Fi network we control to see if our design can obtain a complete list of all Wi-Fi devices in this network. Figure 2(a) shows the list of MAC addresses of Wi-Fi devices connected to our network. We obtain this list from the access point and it serves as the ground truth. There are five smartphones, Samsung Galaxy S7, S8, and S20 and Huawei Y6 and Y7 phones. There are also two tablets, a Microsoft Surface pro 5 and a Samsung Galaxy Tab E.

We now inject fake beacon frames that claim that all of these devices have packets buffered for them at the access point. Note that the attacker does not have access to the above list of devices and he/she wants to find them. Figure 2(b) shows the Wi-Fi traffic exchange captured using Wireshark [6]. The first packet is the fake beacon packet we inject. The Wi-Fi devices think that this packet is actually coming from their access point and respond to it. The next packets in the Wireshark trace are the null packets sent by Wi-Fi devices. The null packets indicate that a device is now awake and ready to receive packets. There is a power saving flag in null packets that client devices can set or unset. When set to 0, it indicates that the device that transmitted this packet will not go to sleep and can receive packets. Note that for better visual presentation we have filtered out other traffic in Figure 2(b). As can be seen in the figure, the attacker can discover the MAC address of all 7 devices in the target network. In this example, one fake beacon could wake up all devices. However, since Wi-Fi devices are in the sleep mode most of the time, we send multiple fake beacons to make sure all Wi-Fi devices receive these packets. In practice, if the attacker sends 10 to 100 beacons per second, all devices respond in at most a few seconds.

It is worth mentioning that this device discovery technique does not affect the normal operation of the target Wi-Fi network. For example in this experiment, we make a phone call over the Internet and we noticed no impact on the call. The only side effect of this procedure is that it prevents Wi-Fi devices from going back to sleep for the duration of the device discovering step which lasts for at most a few seconds.

[2]FF indicates 8 devices. The attacker can test for more devices with longer bitmaps.

4.2 Measuring Time of Flight

After finding the MAC address of target devices, the Wi-Peep attacker performs Time of Flight (ToF) measurements to localize the targets. Past work [36] has shown that we can measure accurate ToF using the Channel State Information (CSI) measured across multiple Wi-Fi channels at both devices. However, we do not expect the target device to share its measured CSI values or switch channels synchronously.

Our approach relies on 802.11 acknowledgments to measure time-of-flight (ToF) between the attacker and the target device. Once we measure ToF, we convert it into distance by multiplying by the speed of light. At a high level, the attacker generates fake 802.11 packets intended for each device identified in the step above. The attacker timestamps packet transmission and ACK reception, and computes the delay between them. The delay corresponds to the round trip time-of-flight and the SIFS (delay between packet reception and ACK transmission defined by the 802.11 standard). SIFS is mandated to be constant (e.g. 10 μs in the 2.4 GHz band). Therefore, we can subtract SIFS from the delay to get a ToF estimate. We delve deeper into this design below.

Generating Fake Wi-Fi Packets: Recall, we do not have access to the target Wi-Fi network. The network may be encrypted. In such cases, how do we generate packets that elicit an acknowledgment from the target device? We rely on a recently proposed method called Polite Wi-Fi [1]. We can send fake and unencrypted packets to a Wi-Fi device and it responds with an ACK. We only need to set the MAC address of the target device and the rest of the fields can be fake. We inject this raw packet. Surprisingly, the target device acknowledges receiving this packet. In Section 5, we describe how we use fake beacon frames to keep target devices awake during the attack so that they continue sending ACKs. Besides Polite Wi-Fi, one can also utilize a similar technique that uses probe response packets instead to generate ACKs from target devices [29].

Estimating Time-of-Flight: The attack sends multiple packets to each target as it passes by. Let us say the attacker measures T_i^j as the delay between packet transmission and ACK reception. T_i^j denotes the i^{th} measurement from client j. Then, T_i^j depends on

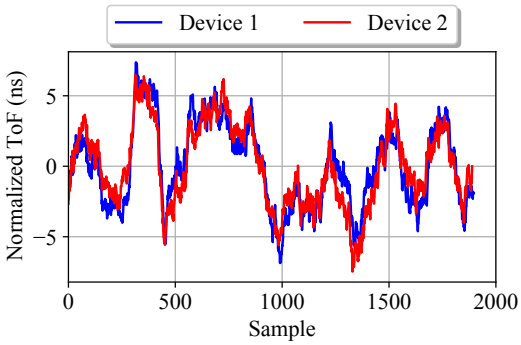


Figure 3: Measuring ToF on two independent ESP32s.

the location (x_j, y_j) of the target:

$$T_i^j = \frac{\sqrt{(X_i - x_j)^2 + (Y_i - y_j)^2}}{c} + \overline{SIFS^j} + \epsilon_i^j \quad (1)$$

where (X_i, Y_i) is the location of the attacker when measuring i, c is the speed of light, and $\overline{SIFS}^j$ is the SIFS delay for the target device. We will soon describe why we expect the SIFS delay to vary per device. Finally, ϵ_i^j is the measurement noise. This measurement noise can have different root causes such as multipath propagation. We describe some of them here:

(i) Per-Device SIFS Variation: Although SIFS is defined by the IEEE 802.11 standard to be 10 μs, the standard allows a deviation of $\pm 1\mu s$. Recent studies [12, 24] and our independent measurements confirm that Wi-Fi devices deviate from the defined SIFS time which results in tens or even hundreds of meters of inaccuracy when using time of flight to estimate distance ($1\mu s$ delay corresponds to 300 m in distance). Moreover, our measurements show that some devices use an incorrect SIFS such as 8 μs (perhaps to achieve a slightly higher throughput). Some devices use 16 μs which is the SIFS defined for the 5 GHZ spectrum probably to simplify their chip design (i.e., same SIFS for both bands). As a result, to find the time of flight, simply subtracting 10 μs does not work. Some systems [24, 25] calibrate for each device to find the actual SIFS before they can use them for localization which is not a practical option in an attack.

(ii) SIFS Jitter: We observe that this unknown per-device SIFS varies slightly over time. This adds some noise to our ToF measurements. The other source of noise is the attacking device itself. Since the attacking device measures the time between a packet and its ACK, its clock frequency determines the accuracy of measurements. Moreover, operating systems processing overheads such as buffering and function calls adds more noise to our measurements.

Next, we characterise how much of this inaccuracy is caused by the attacking device and how much by the target device. We implement the ToF measurement mechanism on an ESP32 Wi-Fi module which runs a real-time operating system. As soon as it detects a packet or an ACK it records the time (i.e., CPU cycle). We describe our implementation in more detail in Section 5. We send fake 802.11 packets to a Microsoft Surface Pro tablet and measure the ToF on two independent ESP32 modules to find the root cause of the noise in measurements. Note that the attack requires only one measuring device. We use two devices only in this experiment for assessing the accuracy of our technique.

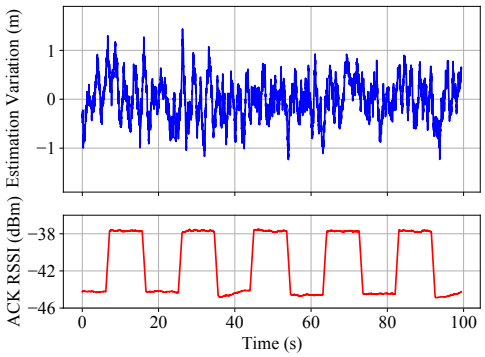


Figure 4: Robustness to RSSI changes

Figure 3 shows the normalized ToF measurements performed by the two ESP32 modules. Since our goal is to evaluate the variation in measuring ToF, we subtract the average of raw values (that include an unknown SIFS and a round-trip time of flight) from raw values measured by each device. Each point in this plot is the average of 100 ToF measurements (i.e., packet-ACK pairs). Since the target and attacking devices are stationary, the measured value should not change over time. However, due to the mentioned reasons, there will be some noise in our measurements. The figure shows that the values reported by both ESP32 devices fluctuate by several nanoseconds. The interesting observation is that both lines follow roughly the same pattern. Since the two ESP32 devices measure the values independently, it is unlikely that they experience correlated errors. So, we conclude that most of the observed variation is caused by the target device and not the measuring (attacking) device. We later evaluate the effect of these variations on localization performance.

(iii) ToF Variance with Signal Strength: Next, we evaluate the effect of the signal strength of Wi-Fi packets on the accuracy of measuring the time of flight. Previous studies [36] report that the signal strength of a packet may impact the time a Wi-Fi device needs to detect it. To see if ESP32 suffers from such an issue, we repeat the last experiment except that the target device changes its transmission power by 7 dB every 10 seconds. As a result, the signal strength of the ACKs received from the target device changes 5 times. Similar to the last experiment, we calculate the normalized ToF. Then, we convert it to distance by multiplying it by the speed of light. Figure 4 shows the variation in estimated distance over time and the Received Signal Strength Indicator (RSSI) of the received ACKs. Each point is the average of 100 measurements. Interestingly, we observe that despite a 5x change in the strength of the signal there is no change in the ToF measurements other than expected variation discussed before. Therefore, our measurement technique is robust to changes of signal strength. In Section 6.3, we test the robustness with a much wider range of RSSIs.

4.3 Localizing Targets

We are given a sequence of noisy ToF measurements, $\{T_i^j\}_{i=1}^N$, for each target device j. We need to estimate the client location from these noisy measurements. To begin with, we observe that the noise can be classified into two categories: (a) Structured per-device error: SIFS variation occurs per device and is constant for that device across measurements, (b) Random device-independent errors:

SIFS jitter and multipath occur randomly and are not fixed for a given device. To enable accurate localization of targets, we need a mechanism to deal with these two types of noise.

First, let us consider the per-device SIFS variation. Note that, the per-device variation in SIFS is a constant for the target device and adds (or substracts) a constant *across all the ToF measurements* for a given device. Since this is a constant value that gets added to all the measurements, we can explicitly model this error as a hidden variable in our design. Theoretically, it adds one variable to our system of equations, which means we need at least one additional (independent) measurement to estimate this. In our design, the collection of one additional measurement is low-cost and hence, this variable can be embedded into our formulation.

Next, we consider the effect of multipath. Objects in the environment reflect the signal from the target device and corrupt Wi-Peep's time-of-flight measurements. This behavior is more prominent when the direct path is blocked and a reflected path may be more prominent than the direct path. In our setting, both direct path and the reflected path have to pass through walls, but additional blockages along the direct path increase the deviation of time-of-flight measurements. Past work, such as [4, 39], have observed that the direct path is stable across measurements in space, while the reflected paths vary widely. For instance, [39] showed that even with a small wavelength-scale motion, reflected paths are inconsistent and can be identified and removed.

In Wi-Peep, the motion of the drone or a passerby is at a much-larger scale and includes hundreds of measurements (due to the low overhead of our design and millisecond-level quick measurements). Therefore, motivated by past work and the spatial diversity of our measurements, we take *wisdom of the crowd approach*. Even though each of our measurements is impacted by multipath reflections, our formulation spanning hundreds of measurements re-inforces the direct path and eliminates the reflected paths. Specifically, the direct path appears in a large fraction of our measurements consistently at the same location and gets enforced. The reflected paths depend on the location of the drone and are local or inconsistent in time. Across measurements, they appear to emerge from different locations and hence get averaged out. Finally, note that we combine the different measurements across time non-coherently, i.e., we do not assume or require phase synchronization across measurements.

Based on the intuition above, we can formulate an optimization problem to identify a location estimate for target device j – $\hat{x_j}, \hat{y_j}$:

$$(\hat{x_j}, \hat{y_j}) = \underset{(x_j, y_j, \overline{SIFS}^j)}{\arg\min} ||T_i^j - \frac{\sqrt{(X_i - x_j)^2 + (Y_i - y_j)^2}}{c} - \overline{SIFS}^j||^2 \quad (2)$$

To solve this optimization problem, we use a grid search approach. This methodology works well for this attack because the target building covers a limited area. For example, if the area of a building is 100 m^2 and we can divide this area into 1000 squares of 0.1 m^2. Therefore, we define a grid with multiple possible *candidate* locations. For each candidate location, we first optimize for $\overline{SIFS}^j$ by plugging the values of x_j, y_j into Eq. 2. Then, we compute the error for this candidate location by plugging in the estimated value of SIFS into this equation. This error gives us an estimate of the fit achieved by the current candidate location with respect to our measurements. We repeat this process for all candidates locations to find the optimal location of the target device. Note that in 3D space, the complexity of grid search approach does not increase substantially. This is because in the z dimension we only test possible heights (i.e., first floor, second floor, etc) based on the floors in the target building to reduce the complexity of the space. In addition, a big advantage of our computation is that it is embarrassingly parallel. Therefore, it can be easily parallelized to significantly speed up computation if needed. In fact, our implementation is a multi-threaded program that uses as many CPU cores as possible to improve the performance.

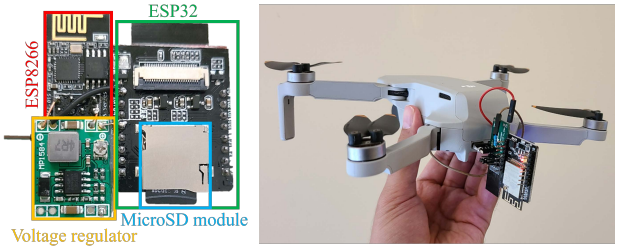


Figure 5: Hardware implementation of Wi-Peep.

We note a few points about this optimization approach. First, to get an accurate location, just three measurements are needed. In our case, due to an extra variable (SIFS), we need an additional measurement. However, we have significant noise in the system as shown in Fig. 3. In addition, multipath causes sporadic errors in our timing estimates. Therefore, we use many more measurements (nearly 100 per device). The ability of Wi-Peep to generate many packets from the target in a short duration of time is very useful. Second, our grid search approach works well for small locations like houses and buildings, and does not require a lot of compute power. For really large locations, we could optimize this further using approaches like gradient descent. In Section 7, we evaluate the effectiveness of this technique in finding the location of different targets in a multi-level house.

5 SYSTEM DESIGN

We aim to demonstrate that it is possible to administer this attack surreptitiously and at a low cost to the attacker. We discuss our design decisions below.

Off-the-shelf Drone: We use an ultra-light DJI mini 2 drone [7] that weights under 250 g. The drone is so light that it does not need a license to operate in many countries. In addition, light drones are desirable for this attack since they generate less noise and they are less noticeable.

Measuring Equipment: We need the measuring equipment to have a small form factor, low weight, and low power consumption. Our equipment, shown in Fig. 5, contains three main modules: ESP32 and ESP8266 low-power Wi-Fi modules and a voltage regulator. These components weigh about 10 g. Since the weight of the drone is 239 g, the total weight is still under 250 g, requiring no license to operate the drone. Although we use a small drone, the weight of the measuring equipment is so low that it has no noticeable impact on the flight operation of the drone.

Generating Traffic: Our design utilizes an ESP8266 Wi-Fi module [10] that injects fake 802.11 packets to target devices. Since battery-operated Wi-Fi devices implement the 802.11 power saving mechanism, they are typically in the sleep mode. As a result, they do not receive our fake packets. To solve this issue, we program the ESP8266 module to send fake beacon frames to the target devices as well as fake packets. In the fake beacon, we set the TIM bitmap to 1 so that the target devices think they must stay up to receive buffered packets from their AP. Since Wi-Fi devices go back to sleep after a while if they do not receive any packet from the AP, we keep sending these fake beacons to target devices to keep them awake during the attack. Specifically, in our implementation with alternate between one fake packet and one fake beacon to make sure the target device does not go back to sleep. For more details, please refer to Section 4.1 where we use a similar technique to discover target devices.

Attack Implementation: We use an ESP32 Wi-Fi module [9] to receive injected packets and ACKs sent by targets device to measure the time of flight. Injecting fake packets and measuring the ToF can potentially be implemented on a single Wi-Fi module but due to some limitations of the API, we use two Wi-Fi modules instead. We need to know the exact time of departure of fake packets to start our ToF timer, however the ESP32 API does not provide this information. Therefore, we use an ESP32 dedicated to just receiving Wi-Fi packets. When it receives a fake packet it records the time (i.e., the current CPU cycle) and when it receives the corresponding ACK from a target device it records the time again. It then subtracts the timestamps to estimate the ToF. The ESP32 module has a 240 MHz clock that allows timestamping with an accuracy of a few nano seconds. We measure the round-trip ToF which doubles the accuracy. Note that the target device can have any clock rate. It is the attacking device that needs accurate timestamping. High clock rates are becoming very common because modern WiFi chipsets support 160 MHz channels, therefore their clock must be at least double that frequency (i.e., 320 MHz). We expect that future WiFi chipsets improve the accuracy of this attack.

The implementation of ToF measurement on the ESP32 module is very sensitive. We need to perform two tasks that can be conflicting. One task measures ToF and the other task records these measurements to a micro SD memory card. On one hand, measuring ToF is very sensitive and must be ultra light-weight in terms of the code it runs. If the measurement procedure is slow, it makes the measurements inaccurate. On the other hand, writing measured values to a memory card is extremely slow. We may not finish writing by the time the next packet arrives. We utilize the two processing cores on ESP32 to solve this issue. Our program runs two threads for measuring and writing the results, each pinned to separate cores. The measurement thread writes to a shared buffer and the writer thread writes this data to the memory card. This allows the measurement thread to exit the interrupt handler (that handles incoming packets) quickly.

Logging Drone Location: In addition to the ToF data, we need to record the location of the drone during measurements. This information is needed by our localization algorithm and recorded on the drone automatically. The drone logs its location information, received from its GPS, every 100 ms. This information includes the lateral position and the altitude above the take off location (and sea level). The drone sends its location information wirelessly to the phone used to control it. However, we need to synchronize the drone data with our ToF measurements performed on the ESP32. To do so, we have soldered a wire to the power port of one of drone's motors and connected it to one of the ESP32 digital input pins. When the motor starts the logic level of this pin starts oscillating between LOW and HIGH (because it is a DC motor) and ESP32 can detect this change. As soon as this signal is received, ESP32 starts measuring. Similarly, when the drone lands at the end of a measurement, ESP32 detects it and stops capturing and flushes all remaining data to the memory card. We note that it is not necessary to solder a wire to find out the start and end of a flight. The attacker can manually start and stop ESP32. We implemented this modification for convenience because we wanted to run many experiments. Finally, the attacker reads the memory card and performs Wi-Peep's localization procedure.

Power: To keep the equipment lightweight, we avoid adding batteries to power the equipment. Instead, we use the drone's battery to power our circuits. We connect a wire from the positive terminal of the drone's battery to the Vdd input of our circuit. We also add a USB-C connector to our design so that we can physically attach the measurement equipment to the drone.[3] In addition, we use the ground terminal (GND) of the USB port to complete the power circuit.[4] We cannot use the drone's battery directly because its voltage is between 7-8 Volts. However, both ESP32 and ESP8266 modules require 3.3 V input. We use an MP1584EN DC-DC bulk converter [26] to efficiently convert 7-8 V to 3.3 V. We measure the power draw of the entire circuit and we found that it impacts the flight time of the drone by only 10%. This means we can fly the drone with the measuring equipment for about 20-25 minutes. Our tests show that an attacker requires only a few minutes to complete the attack. We conducted experiments in different wind conditions from calm to 22 km/h (14 mph), gusting up to 32 km/h (20 mph). Although the drone had vibration and random movements in windy conditions, we noticed no impact on the results.

6 MICROBENCHMARK EVALUATIONS

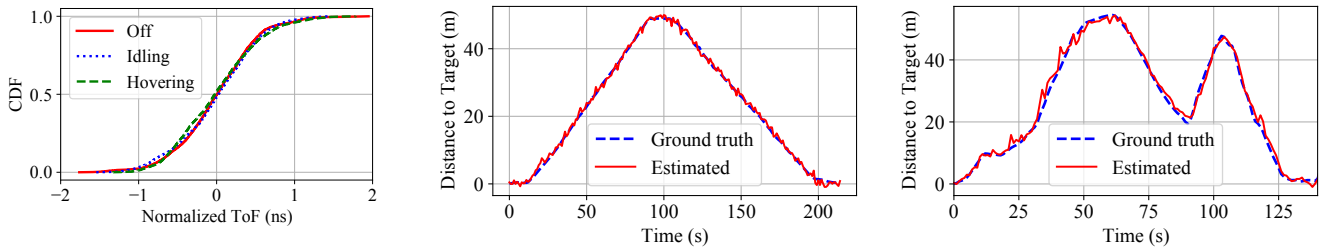
We present some microbenchmarks for Wi-Peep below.

6.1 Accuracy of Drone's Positioning System

The accuracy of Wi-Peep depends on the accuracy of the positioning system of the drone as formulated in Equation 2. Wi-Peep relies on the distance between the drone and candidate locations which is driven from the drone's Global Navigation Satellite System (GNSS). Wi-Peep can achieve high localization accuracy only if the GNSS data is accurate. To evaluate the accuracy of the drone's positioning system, we conduct a series of experiments.

We place the drone at 7 locations and record the coordinates reported by its positioning system. Some locations are 500 meters away from other locations. Since GPS performance varies over time, we use the Randomized Multiple Interleave Trial (RMIT) technique

[3]The DJI mini 2 drone has a USB-C port for retrieving its data.
[4]We cannot use the USB-C port to receive power from the drone because it is passive and does not provide power. We can only use its ground pin



(a) Impact of drone/propellers on ToF **(b) Vertical flight results** **(c) Horizontal flight results**

Figure 6: Wi-Peep microbenchmark results

previously used in variable environments such as Wi-Fi networks and cloud computing systems [2, 3]. In this technique, in every trial one of the 7 locations is chosen randomly and the experiment is repeated multiple times. For each location, we record about 1000 measurements. We perform the measurements over one week in different meteorological conditions from sunny to overcast.

For every location, we find the ground truth latitude and longitude from Google Maps. Although these satellite maps have some errors, we believe that these errors are irrelevant for Wi-Peep. This is because after Wi-Peep reveals the location of a target device, the attacker must use a satellite map to find where the obtained latitude and longitude for the target device is actually located. As a result, the important metric is the relative error between the drone's GNSS and the map. For example, if both the drone's GNSS and a map have the same error (e.g., both have a 1 m deviation to east) it does not create any error for Wi-Peep. As a result, we only care about the agreement between the GNSS and the map used for localization.

Figure 7 plots the CDF of the drone's GNSS error (relative to Google Maps). We also plot the CDF of all measurements across all locations. We can see that the median error is 75 cm for all measurements. In these experiments, the drone hovers above a given location at low altitudes to make sure it is actually at that given location. At these lower altitudes, the GPS signal from some satellites might be partially blocked or the signal might be reflected from nearby buildings which increases the error. In contrast, during our localization experiments presented later, the drone flies at a much higher altitude (above a building) with perfect line-of-sight to all GNSS satellites. Therefore, we believe that the GNSS accuracy is even higher during the attack.

One might think that the accuracy of GPS should be much worse than 75 cm. This is true, but the reason the drone could achieve a higher accuracy is that it actually uses multiple GNSS systems. The

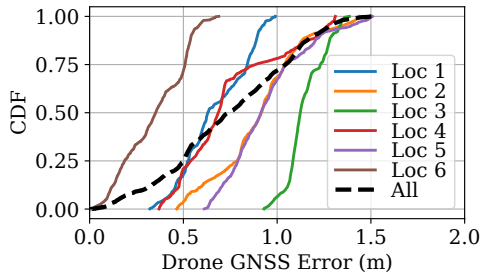


Figure 7: The accuracy of Drone's GNSS system

DJI mini 2 drone we use in our experiments uses a combination of GPS, GLONASS, and GALILEO systems [7]. Instead of the typical 6-8 GPS satellites, we observed that the drone locks to up to 28 GNSS satellites during our experiments. Using multiple GNSS systems is becoming typical for modern drones. For example, DJI Mavic 3 and Mini 3 pro use GPS, Galileo, and BeiDou. EVO Nano+ and LITE series use GPS, GLONASS, Galileo. Finally, the Parrot ANAFI drone uses four systems namely, GPS, GALILEO, GLONASS,and BeiDou. Interestingly, there are drones, such as DJI PHANTOM 4 RTK, that can achieve a positioning accuracy of 1.5 cm [8] using the RTK technology. In this technology, the GNSS error is corrected using the data received from a nearby ground station. We believe that using such drones will further improve the accuracy of Wi-Peep.

6.2 Impact of Drone Motion

In existing localization systems, the measuring device is located in a fixed position with no motion. However, we mount the measuring equipment on a drone with fast spinning propellers. Some propellers are just a few centimeters to the Wi-Fi antenna. We conduct an experiment to see if the propeller's motion has any effect on measurements. In this experiment, the drone in on the second floor of a building and the target device is below it on the main floor.

We consider three scenarios. The first scenario is our reference measurement in which the drone is turned off and placed on the floor. Therefore, there is no motion near the measuring equipment. In the second scenario, the drone is still on the ground but its propellers are spinning (i.e., idling). Finally in the third scenario, the drone takes off and hovers at about 1 m from the ground. In this scenario, the propellers are spinning very fast and there is vibration and small movements because of hovering.

Each experiment lasts for 60 seconds. In all experiments, the distance of the drone to the target device remains constant. Therefore, any variation in the ToF measurement is due to noise. In Fig. 6(a), we plot the normalized ToF measurements where we subtract the mean from the measured values and we plot the CDF. As shown in the figure, the CDF obtained form all three scenarios are very similar. This experiment shows that the drone motion does not introduce any additional noise in our ToF measurements.

6.3 Accuracy of ToF Measurements

In the next experiment, we evaluate how accurately a drone can measure the travel time of a signal. We place a drone near a Microsoft Surface pro tablet on the ground. The drone flies vertically to an altitude of 50 meters with a constant speed of about 0.5 m/s

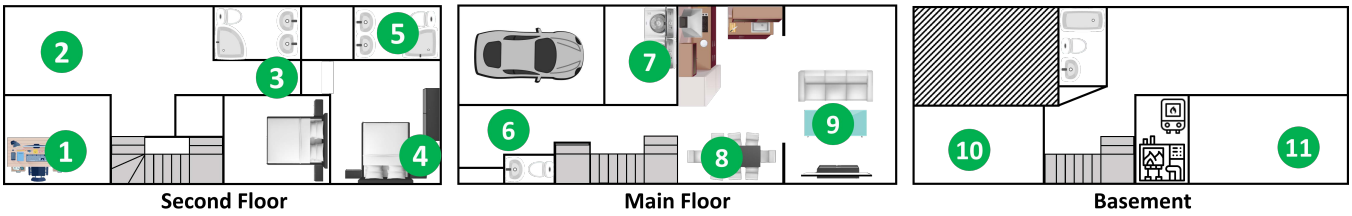


Figure 8: The floor plan of the target building.

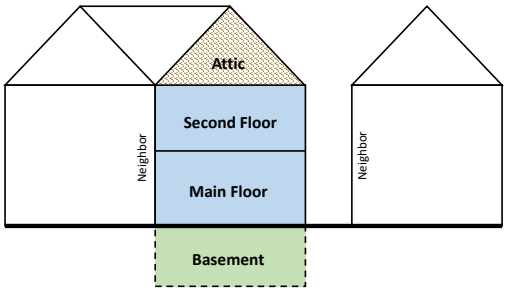


Figure 9: The structure of the target house.

and returns to the ground at the same speed. The drone reports its flying altitude from the take off location which is used as the ground truth. It also measures the signal travel time to the target device and back. The measured time includes an unknown SIFS.

In this particular experiment, we know that the drone and the target device are collocated, therefore the measured distance should be 0 m during the first few seconds. We use this information to calibrate the data. Note that we do not and cannot use such tricks to find the unknown SIFS in the actual localization attack. This method is only used in this microbenchmark experiment. We run a moving average with a window of 1 second over our ToF measurements to smooth out variations in the experiments in this section.

Fig. 6(b) shows the estimated distance to the target device and the ground truth reported by the drone. The distance is 0 m at the beginning when the drone is near the device. Then, the drone climbs to 50 m and comes back to where it took off. We see a near-perfect match between the estimated distance and the ground truth. This confirms that our system works as expected in this simple scenario where the only change is the altitude of the drone.

We run a more complex experiment wherein the drone has lateral movements too. The drone is placed near the target device and flies to an altitude of 15 m. Then it flies in the vicinity up to 30 meters away from the take off location. We evaluate if the distance to the target device can be estimated accurately. The ground truth is extracted from the log file generated by the drone. It reports its coordinates and altitude ten times per seconds so we can calculate its distance to the target device accurately. Fig. 6(c) shows that the estimated distance is very close to the ground truth. These experiments are very promising. However, they are performed in a line-of-sight setting where we have a line of sight path between the drone and target device. We next test how accurately we can localize target devices inside a building.

7 REAL TESTBED EVALUATION

In these experiments, a drone, equipped with Wi-Peep flies over the target building to estimate the location of devices inside the building. The target network is an IEEE 802.11ax WiFi network. We test two target devices, a Microsoft Surface Pro 5 (supports 802.11ac) and a Samsung Galaxy S20 (supports 802.11ax).

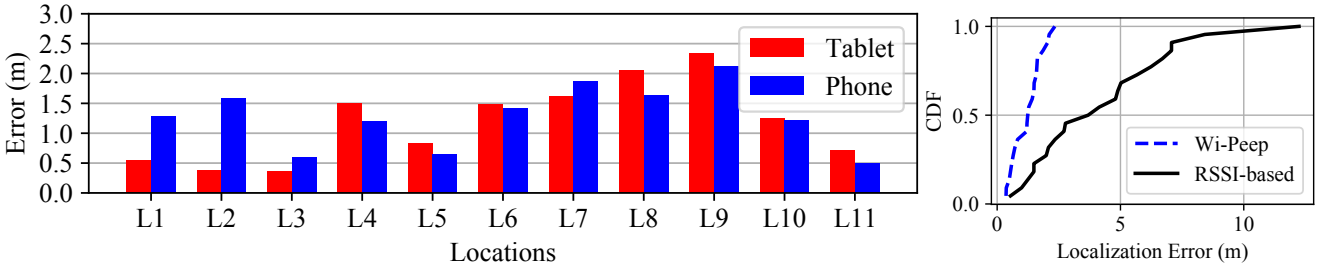
7.1 Testbed

To evaluate the accuracy of Wi-Peep, we run experiments in a house that has three levels (i.e., basement, main floor and second floor). There is also an attic on top of the second floor (mostly empty and filled with a layer of insulation material to prevent heat loss). We place target devices in different locations inside the house. This is a semi-detached house (i.e., connected to another house from one side). We conduct experiments at all floors to evaluate the efficacy of the localization technique when there are many layers of obstacles between the drone and target devices. Figures 8 and 9 show the floor plan of the house and its structure.

The target device is placed on a chair except locations L1, L7, and L8. We now explain each location and describe factors that may impact localization. **L1 - Office:** the target device is placed on a cluttered desk with a 21 inch monitor about 50 cm from the device. **L2 - Family room:** this mostly empty room is above the garage (with a car parked in it). **L3 - Hallway:** this location is between a bedroom and a bathroom. **L4 - Master bedroom:** the target device is near a queen bed which has many metal coils. There is some furniture near the device too. **L5 - Bathroom:** the target device is placed inside the bathroom. A 2 m shower glass is 50 cm away from the device. **L6 - Entrance:** there are a closet, a bathroom, the garage and some furniture near this location. **L7 - Laundry room:** The target device is placed on a dryer machine. There are many metal obstacles near this location: a washing machine and a dryer. The kitchen is behind the wall and the device is very close to a big range hood, stove, cabinets, and a side-by-side fridge. **L8 - Dining room:** there are a dining table, chairs, and some house plants near the target device. Moreover, this location is above the utility room with a furnace, water softener, and many metal air ducts. **L9 - Living room:** there are a sofa, a coffee table, a 42 inch TV, and some other furniture near this location. **L10 - Storage room:** this room is full of (mostly full) boxes up to the ceiling. It is a very cluttered space. **L11 - Basement bedroom** this room is mostly empty.

7.2 Results: Lateral Accuracy

Finding the ground truth locations: Finding the actual coordinates of these locations turned out to be much more difficult than we expected. We cannot use GPS because the error was several meters inside our target building. Finding these locations on satellite maps is also challenging because the satellite was not exactly over the target building when taking the image. Therefore, the height of



(a) Wi-Peep's lateral deviation from the ground truth **(b) Wi-Peep vs RSSI-based**

Figure 10: Lateral accuracy of Wi-Peep in a real target environment.

the building creates a few meters of displacement. Finally, we used Google Earth 3d buildings feature because it compensates for the angle of imaging and height of buildings. We first find the coordinates of the four corners of the target building. We then measure the distance of each location from external walls of the building. Finally, we calculate the coordinates of each target location.

We now evaluate the lateral accuracy of Wi-Peep in finding the location of target devices. In this experiment, we fly a drone over a target house (described in Section 7.1) for about 3 minutes. During this time, the drone sends fake packets to target devices and measures the time of flight. We then apply our localization model (described in Section 4.3) to the collected data to find the location of target devices. We use a Samsung Galaxy S20 smart phone and a Microsoft Surface pro tablet as our target devices. We place each device in all 11 locations described before and we repeat the experiment.

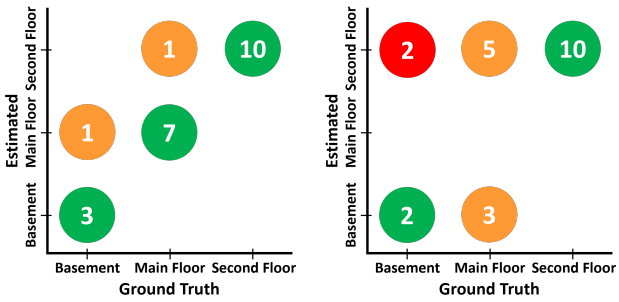
Fig. 10(a) shows the lateral localization error of Wi-Peep in these locations and for the two target devices. The median localization error is 1.26 meters with the max error of 2.3 meters. The accuracy of localization is higher for L1-L5 on the second floor of the building (i.e., error under 1.5 m). This is expected because the path between the drone and the target devices is less blocked for these locations. The localization error increases slightly for L6-L9 on the main floor. These locations are particularly challenging for a few reasons: 1) there is significant amount of RF reflection on the main floor because of big appliances. 2) there is reflection and blockage from the utility room in the basement and 2 Queen size beds on the second floor. Despite these challenging condition, the accuracy of localization is surprisingly good. Finally, the accuracy of the two locations in the basement, L10 and L11, is great despite the significant amount of blockage. The signal strength of ACKs coming from these locations is very weak (i.e., -80 to -90 dBm). However, Wi-Peep can still localize target devices in the basement with a very high accuracy. We believe that the accuracy is higher in the basement compared to the main floor because there are fewer big reflective objects in the basement.

We plot the CDF of lateral localization error for all locations and devices in Fig. 10(b). To compare these numbers with a baseline we have recomputed the locations based on the Received Signal Strength Indicator (RSSI) instead of ToF. In this experiment, we have also collected the RSSI of received ACKs from target devices. Instead of ToF, we use RSSI in our localization model and plot the CDF of lateral error in Fig. 10(b) too. As you can see, when we use RSSI instead of ToF the localization error increases significantly. The median error is 3.9 m when we use RSSI instead of ToF. This shows ToF is a much more robust metric. We cannot compare Wi-Peep with other Wi-Fi localization techniques because they have requirements that prevents us from using them as a localization attack (as explained in Section 3). We note that Wi-Peep's localization error is comparable to the state-of-the-art approaches in indoor localization that rely on antenna arrays and/or co-operation from end devices. For example, in *non-line-of-sight* environments, SpotFi [21] achieves a median error of 1.6 m and a max error of 7 m. Wi-Peep can achieve this accuracy with just coarse grained timestamps and in spite of multiple layers of blockage by leveraging the multiple measurements enabled by our lightweight design.

7.3 Results: Vertical Accuracy

In multi-level buildings, such as our target building, the attacker might need to know on which floor a target device is. For multi-level buildings, our localization model considers candidate target locations on all floors. As a result, instead of just finding the best (x,y) coordinates, it picks the best (x, y, z) point. We segment the value of z to identify the different floors in the house.

Fig. 11(a) shows the accuracy of Wi-Peep in finding the correct floor for the 22 device/location combinations (used in the previous experiment). In the 4 basement experiments, 3 of them where correctly localized as basement and the phone/L11 configuration was



(a) Wi-Peep **(b) RSSI-based localization**

Figure 11: Floor-identification accuracy of Wi-Peep. The number in each circle represents the number of device/location configurations in that category. Green represents correct floor detection. Orange and red represent deviation by one and two floors, respectively.

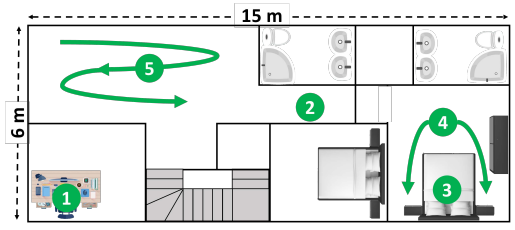
mistakenly reported as main floor. In the 8 main floor experiments, 7 of them are correct and phone/L7 was reported to be on the second floor. L7 is probably the most challenging location on the main floor because it is surrounded by several big appliances. Finally, all 10 experiments on the second floor were reported correctly. In these experiments, Wi-Peep achieves a vertical accuracy of 91%. The 9% mismatch cases are off only by one floor.

Fig. 11(b) shows the performance of RSSI-based localization in detecting the vertical height of targets. Two basement locations are reported to be on the second floor. None of the main floor locations were identified correctly. The results show that except the top floor, the RSSI-based localization technique struggles to find the correct level most of the time.

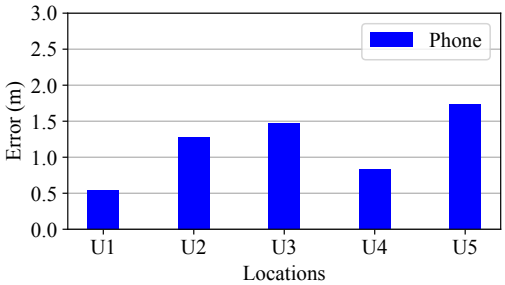
7.4 Results: Mobility

Our evaluation so far has focused on stationary target devices. We now evaluate the performance of Wi-Peep when a user actively uses the target device or when the target device is carried by the user. Fig. 12(a) shows the layout of the experiment. In scenarios 1 and 2 (denoted U1 and U2), a user sits on a chair and holds a smart phone and actively uses the phone (e.g., typing, scrolling, etc). U3 is similar to U1 and U2 except that the user lies on the bed. In U4 and U5 the user walks in the specified room. The phone's screen is off and it is in the user's pocket. This creates a very challenging situation for Wi-Peep since the phone is close to the body. This situation creates significant distortion in the signal coming from the phone both because of mobility and blockage by the body.

Fig. 12(b) shows the lateral localization error of Wi-Peep. For U1-3 the ground truth is the user's location and for U4-5 the ground truth is the center of the room. The figure shows that despite these challenging scenarios, Wi-Peep can localize the target device with high accuracy. The vertical position of the target device (not shown here) is also detected correctly.



(a) Experiment setup



(b) Localization error

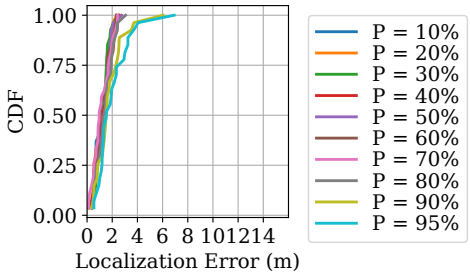
Figure 12: Accuracy in scenarios with mobility.

7.5 Number of Samples vs. Accuracy

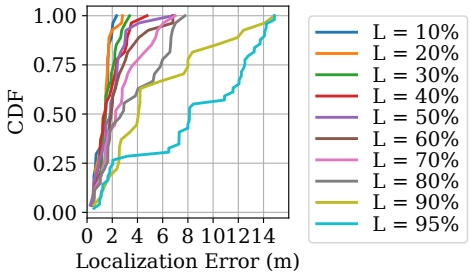
Our previous experiments show that Wi-Peep achieves a very high accuracy in the tested scenarios. In this section, we study how many samples we need and how long the experiment needs to be to achieve such accuracy. In our experiments in Sections 7.2 and 7.3, the drone flies around 3 minutes for each experiment and we send about 100 fake packets per second to a target device. Therefore, each experiment has about 18,000 raw measurements.

The first metric we study is how the accuracy changes as we decrease the number of data points. We uniformly downsample the traces we collected before by a ratio of P. We tested different downsampling factors from 0 (i.e., no downsampling) to 80 % and calculate the accuracy. We repeat this procedure for all 22 device/location stationary settings and 5 scenarios with mobility and plot the CDF of lateral localization error. Fig. 13(a) shows that even when we keep only 20% of our data points (P = 80%), the accuracy is almost identical to that of our original traces. This is an important property for Wi-Peep because it suggests that it can track devices accurately even with a fraction of samples. This experiment also shows that Wi-Peep can scale to numerous devices simultaneously by leveraging the extra packets to attack other devices.

Next, we study the impact of experiment duration which determines the area the drone covers during each experiment. We truncate our traces by a factor of L and recalculate the accuracy. Fig. 13(b) shows the CDF of lateral accuracy. Unlike downsampling, reducing the duration of the experiment significantly increases localization error. The plot shows that if we lose 30% of our traces the accuracy is still reasonably good but anything beyond that probably renders the localization useless. This experiment shows that



(a) Downsampling



(b) Truncating

Figure 13: The impact of downsampling and reducing the length of the trace on the accuracy of Wi-Peep. Dashed lines show 0.9 and 0.95 CDF.

the spatial diversity is critical for Wi-Peep. Together, these experiments indicate that an experiment where the drone moves faster but covers the same span is optimal from an attacker's perspective.

8 SUGGESTED DEFENSE MECHANISM

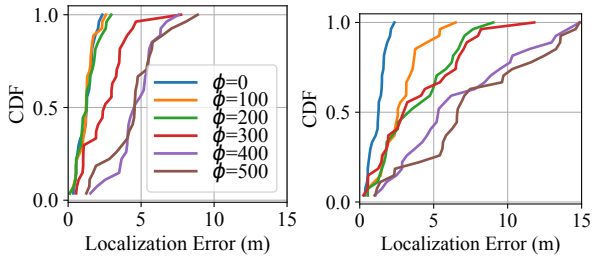
Our goal in this paper is to highlight vulnerabilities in the 802.11 protocol that lead to location leakage from unsuspecting devices. In this section, we discuss potential solutions for the Wi-Peep attack. At a high level, to disrupt this attack, a Wi-Fi device needs to make the time of flight measurements ineffective. The ToF measurement is based on responding to fake 802.11 packets by sending ACKs. Past work [1] argues that preventing the response to fake 802.11 packets is impossible. This is because a Wi-Fi device needs to check the encryption of packets to see if they are authentic. However, the device only has 10 μs (i.e., SIFS) for checking the encryption which takes at least an order of magnitude longer in practice [20, 23, 30]. Therefore, detecting these fake packets in a timely manner is very challenging if not impossible.

One may wonder if WiFi devices can simply monitor for bursts of fake packets to detect this attack. Although the attack can be detected, the physical layer still sends back ACKs and the target device cannot stop the attack. In addition, the fake beacon frames we send to keep target devices awake are difficult to be identified, because they are identical to real beacons since beacons do not and cannot have encryption. Note that WPA3 has introduced beacon protection which allows WiFi devices to detect fake beacons. However this optional feature has not been implemented on any chipset yet and it may never become a popular feature because of lack of backward compatibility with billions of existing WiFi devices[31].

Since Polite WiFi cannot be prevented, we propose a different solution that targets the ToF measurements directly. Since the attacker measures the time between a packet and its ACK, can we randomize the time between packets and ACKs to make ToF measurements ineffective? A device can potentially add a small noise to SIFS to confuse the attacker. Although the IEEE 802.11 standard defines SIFS to be 10 μs and 16 μs in the 2.4 and 5 GHz bands, it allows a ± 1 μs deviation from this time. While this solution appears simple, it is challenging to implement in practice because it requires hardware and/or firmware modifications which can only be implemented by hardware vendors. Therefore, this solution is most suitable for designing future Wi-Fi devices.

To evaluate the efficacy of this defense, we emulate the solution on our ToF measurements. We use the tablet and phone measurements in 11 stationary and 5 mobile scenarios. We add noise to our ToF measurements as if the noise were added by the target device. This is virtually the same as implementing the solution on target devices. The first solution we evaluate is adding uniform noise. For every ToF measurement, we draw a random number from the range $[-\phi, \phi]$ and add it to the time of flight.

Fig. 14(a) shows the localization accuracy when we change ϕ from 0 (i.e., no solution) to 500 ns for all 27 device/location combinations. Since Wi-Fi devices might have some inaccuracies in their SIFS, we do not use the entire 1000 ns permitted deviation. The figure shows that when the noise is up to 200 ns, we do not see any significant loss of accuracy. This is reasonable since Wi-Peep's localization model is robust to some noise as we have seen before.



(a) $[-\phi, \phi]$ **noise every packet** **(b)** $[-\phi, \phi]$ **noise every 5 s**

Figure 14: Potential defence against Wi-Peep.

However, when the noise is more that 400 ns the median of the localization error is increased to more than 4 m.

Since Wi-Peep uses the average of ToF measurements over one second, it is robust to a uniform noise to some extent. We present a more effective solution for the localization attack. Recall, a drone flies to different locations to measure the ToF. Instead of a random noise for every packet, we can keep the noise constant for a few seconds and then change it to another value. If the target device continues this procedure, it can pretend it is moving which introduces significant errors.

In the next experiment, the target device draws a random number in the range $[-\phi, \phi]$ and adds it to all SIFS for 5 seconds. Then it draws another random number and uses it for 5 seconds. The target device continues this procedure for the entire experiment. The 5 seconds interval was chosen because it gives the drone enough time to move to a different location but it is not too long that a localization method can infer much information during this time. Fig. 14(b) shows the localization error in this experiment. As you can see, this approach significantly increases the localization error. With ϕ set to 200 ns, the error goes up to 9 meters. Overall, the second method introduces more error and it might be more lightweight solution because Wi-Fi devices do not need to change the noise for every packet. We recommend Wi-Fi chipset vendors to consider these solutions to patch their Wi-Fi devices against non-cooperative localization attacks.

9 ETHICAL CONSIDERATIONS

The house and WiFi devices used in our experiments are owned and controlled by the authors. In order to expedite mitigating the attack presented in this paper, we have started engagements with WiFi access point and chipset manufacturers. We have also discussed our experiment protocol with our institution's IRB and the IRB determined that this does not constitute human subject research.

10 CONCLUSIONS

In this paper, we introduce a new localization attack against Wi-Fi devices called Wi-Peep that does not require any voluntary cooperation from Wi-Fi devices. The required hardware for this attack is small, light weight, and very cheap. Our implementation of the attacking device can be carried by an ultra-light drone. Our evaluation of the localization accuracy of Wi-Peep in a real environment shows that target devices can be localized with a median accuracy of 1.2 m. We present and evaluate a potential solution that can secure future Wi-Fi devices against this attack.

ACKNOWLEDGMENTS

We thank anonymous reviewers and our shepherd for providing valuable and insightful feedback on this paper. We acknowledge the support of the Natural Sciences and Engineering Research Council of Canada (NSERC).

REFERENCES

[1] Ali Abedi and Omid Abari. 2020. WiFi Says" Hi!" Back to Strangers!. In *Proceedings of the 19th ACM Workshop on Hot Topics in Networks*. 132–138.
[2] Ali Abedi and Tim Brecht. 2017. Conducting Repeatable Experiments in Highly Variable Cloud Computing Environments. In *ICPE*.
[3] Ali Abedi, Andrew Heard, and Tim Brecht. 2015. Conducting Repeatable Experiments and Fair Comparisons Using 802.11n MIMO Networks. *SIGOPS Operating Systems Review* (2015).
[4] Fadel Adib, Zachary Kabelac, and Dina Katabi. 2015. Multi-person Localization via RF Body Reflections *(NSDI)*.
[5] Victor Bahl and Venkat Padmanabhan. 2000. RADAR: An In-Building RF-based User Location and Tracking System *(INFOCOM)*.
[6] Gerald Combs. 2020. *Wireshark*. https://www.wireshark.org/.
[7] DJI 2022. *DJI Mini 2*. DJI. https://www.dji.com/ca/mini-2/specs.
[8] DJI 2022. *PHANTOM 4 RTK*. DJI. https://www.dji.com/ca/phantom-4-rtk/info.
[9] Espressif Systems 2019. *ESP32 datasheet*. Espressif Systems. https://www.espressif.com/sites/default/files/documentation/\esp32_datasheet_en.pdf.
[10] Espressif Systems 2020. *ESP8266 datasheet*. Espressif Systems. https://www.espressif.com/sites/default/files/documentation/0a-esp8266ex_datasheet_en.pdf.
[11] Julien Freudiger. 2015. How Talkative is Your Mobile Device? An Experimental Study of Wi-Fi Probe Requests. In *Proceedings of the 8th ACM Conference on Security and Privacy in Wireless and Mobile Networks (ACM WiSec '15)*.
[12] Domenico Giustiniano, Theodoros Bourchas, Maciej Bednarek, and Vincent Lenders. 2015. Deep Inspection of the Noise in WiFi Time-of-Flight Echo Techniques. In *MSWiM*. 5–12.
[13] Jon Gjengset, Jie Xiong, Graeme McPhillips, and Kyle Jamieson. 2014. Phaser: Enabling Phased Array Signal Processing on Commodity Wi-Fi Access Points. *MobiCom* (2014).
[14] Omar Hashem, Moustafa Youssef, and Khaled A. Harras. 2020. WiNar: RTT-based Sub-meter Indoor Localization using Commercial Devices. In *IEEE International Conference on Pervasive Computing and Communications (PerCom)*.
[15] Suining He and S.-H. Gary Chan. 2016. Wi-Fi Fingerprint-Based Indoor Positioning: Recent Advances and Comparisons. *IEEE Communications Surveys Tutorials* 18, 1 (2016), 466–490. https://doi.org/10.1109/COMST.2015.2464084
[16] Baik Hoh, Marco Gruteser, Hui Xiong, and Ansaf Alrabady. 2007. Preserving Privacy in Gps Traces via Uncertainty-Aware Path Cloaking *(ACM CCS)*.
[17] Berthold K.P. Horn. 2020. Doubling the Accuracy of Indoor Positioning: Frequency Diversity. *Sensors* (2020).
[18] Mohamed Ibrahim, Hansi Liu, Minitha Jawahar, Viet Nguyen, Marco Gruteser, Richard Howard, Bo Yu, and Fan Bai. 2018. Verification: Accuracy Evaluation of WiFi Fine Time Measurements on an Open Platform. In *Annual International Conference on Mobile Computing and Networking (ACM MobiCom)*.
[19] Tao Jiang, Helen J. Wang, and Yih-Chun Hu. 2007. Preserving Location Privacy in Wireless Lans *(ACM MobiSys)*.
[20] S. S. Kolahi and A. A. Almatrook. 2017. Impact of security on bandwidth and latency in IEEE 802.11ac client-to-server WLAN. In *2017 Ninth International Conference on Ubiquitous and Future Networks (ICUFN)*. 893–897.
[21] Manikanta Kotaru, Kiran Joshi, Dinesh Bharadia, and Sachin Katti. 2015. SpotFi: Decimeter Level Localization Using Wi-Fi *(SIGCOMM)*.
[22] Swarun Kumar, Stephanie Gil, Dina Katabi, and Daniela Rus. 2014. Accurate Indoor Localization with Zero Start-up Cost *(MobiCom)*.
[23] P. Li, S. S. Kolahi, M. Safdari, and M. Argawe. 2011. Effect of WPA2 Security on IEEE 802.11n Bandwidth and Round Trip Time in Peer-Peer Wireless Local Area Networks. In *2011 IEEE Workshops of International Conference on Advanced Information Networking and Applications*. 777–782.
[24] Andreas Marcaletti, Maurizio Rea, Domenico Giustiniano, Vincent Lenders, and Aymen Fakhreddine. 2014. Filtering Noisy 802.11 Time-of-Flight Ranging Measurements. In *CoNEXT*. 13–20.
[25] Alex T. Mariakakis, Souvik Sen, Jeongkeun Lee, and Kyu-Han Kim. 2014. SAIL: Single Access Point-Based Indoor Localization. In *Annual International Conference on Mobile Systems, Applications, and Services (ACM MobiSys)*.
[26] Monolithic Power Systems Inc. 2011. *MP1584 Step-Down Converter*. Monolithic Power Systems Inc. https://www.monolithicpower.com/en/documentview/productdocument/index/version/2/document_type/Datasheet/lang/en/sku/MP1584EN-LF-Z/document_id/204/.
[27] Michał Nowicki and Jan Wietrzykowski. 2017. Low-effort place recognition with WiFi fingerprints using deep learning. In *International Conference Automation*. Springer, 575–584.
[28] Anshul Rai, Krishna Kant Chintalapudi, Venkata N. Padmanabhan, and Rijurekha Sen. 2012. Zee: Zero-effort Crowdsourcing for Indoor Localization *(MobiCom)*.
[29] Maurizio Rea and Domenico Giustiniano. 2021. Location-aware Wireless Resource Allocation in Industrial-like Environment. *IEEE Transactions on Mobile Computing* (2021).
[30] Mohammad Saleh, Jaafar Gaber, and Maxim Wack. 2017. Sensor Networks Applications Performance Measures for IEEE802.11n WiFi Security Protocols. In *Proceedings of the International Conference on Future Networks and Distributed Systems (ICFNDS '17)*. https://doi.org/10.1145/3102304.3102335
[31] Domien Schepers, Aanjhan Ranganathan, and Mathy Vanhoef. 2022. On the Robustness of Wi-Fi Deauthentication Countermeasures *(WiSec '22)*. 245–256.
[32] B. Schilit, J. Hong, and M. Gruteser. 2003. Wireless location privacy protection. *IEEE Computer* (2003). https://doi.org/10.1109/MC.2003.1250896
[33] Reza Shokri, George Theodorakopoulos, Panos Papadimitratos, Ehsan Kazemi, and Jean-Pierre Hubaux. 2014. Hiding in the Mobile Crowd: LocationPrivacy through Collaboration. *IEEE Transactions on Dependable and Secure Computing* (2014).
[34] Reza Shokri, George Theodorakopoulos, Carmela Troncoso, Jean-Pierre Hubaux, and Jean-Yves Le Boudec. 2012. Protecting Location Privacy: Optimal Strategy against Localization Attacks *(ACM CCS)*.
[35] Ivan Vasilevski, Dobre Blazhevski, Veno Pachovski, and Irena Stojmenovska. 2019. Five Years Later: How Effective Is the MAC Randomization in Practice? The No-at-All Attack. In *ICT Innovations 2019. Big Data Processing and Mining*, Sonja Gievska and Gjorgji Madjarov (Eds.). Springer International Publishing.
[36] Deepak Vasisht, Swarun Kumar, and Dina Katabi. 2016. Decimeter-Level Localization with a Single Wi-Fi Access Point *(NSDI)*.
[37] Jue Wang and Dina Katabi. 2013. Dude, Where's My Card?: RFID Positioning That Works with Multipath and Non-line of Sight *(SIGCOMM)*.
[38] Yaxiong Xie, Jie Xiong, Mo Li, and Kyle Jamieson. 2018. mD-Track: Leveraging Multi-Dimensionality in Passive Indoor Wi-Fi Tracking. *arXiv preprint arXiv:1812.03103* (2018).
[39] Jie Xiong and Kyle Jamieson. 2013. ArrayTrack: A Fine-grained Indoor Location System *(NSDI)*.
[40] Jie Xiong, Kyle Jamieson, and Karthikeyan Sundaresan. 2014. Synchronicity: Pushing the Envelope of Fine-grained Localization with Distributed Mimo. In *HotWireless*.
[41] Jie Xiong, Karthikeyan Sundaresan, and Kyle Jamieson. 2015. ToneTrack: Leveraging Frequency-Agile Radios for Time-Based Indoor Wireless Localization *(MobiCom)*.
[42] Moustafa Youssef and Ashok Agrawala. 2005. The Horus WLAN Location Determination System *(MobiSys)*.
[43] Yanzi Zhu, Zhujun Xiao, Yuxin Chen, Zhijing Li, Max Liu, Ben Y. Zhao, and Haitao Zheng. 2020. Et Tu Alexa? When Commodity WiFi Devices Turn into Adversarial Motion Sensors. *Network and Distributed Systems Security (NDSS) Symposium* (2020).