Secure Crowdsourced Indoor Positioning Systems

Tao Li*, Yimin Chen*, Rui Zhang†, Yanchao Zhang*, and Terri Hedgpeth*
* Arizona State University, † University of Delaware
{tli, ymchen, yczhang, terrih}@asu.edu, ruizhang@udel.edu

***Abstract*—Indoor positioning systems (IPSes) can enable many location-based services in large indoor environments where GPS is not available or reliable. Mobile crowdsourcing is widely advocated as an effective way to construct IPS maps. This paper presents the first systematic study of security issues in crowdsourced WiFi-based IPSes to promote security considerations in designing and deploying crowdsourced IPSes. We identify three attacks on crowdsourced WiFi-based IPSes and propose the corresponding countermeasures. The efficacy of the attacks and also our countermeasures are experimentally validated on a prototype system. The attacks and countermeasures can be easily extended to other crowdsourced IPSes.**

I. Introduction

Indoor positioning systems (IPSes) have received tremendous attention from the academia and industry. IPSes target large indoor environments where GPS signals are unavailable, unreliable, or inadequate. They can actively locate mobile users (devices), provide ambient location context, or navigate them through an indoor venue of interest. They can also enable many location-based services, e.g., location-based proximity advertising inside a shopping mall. There are many competing IPS technologies with great commercial potential. For example, WiFi-based IPS applications are expected to generate revenues up to $2.5 billion by 2020 [1].

A typical WiFi-based IPS depends on the ubiquity of indoor WiFi APs and works in three phases. In the training phase, the IPS collects Received Signal Strength (RSS) fingerprints for all possible indoor positions to build a fingerprint database. In the operating phase, the IPS searches its fingerprint database for the closest match to a submitted RSS fingerprint and then returns the corresponding position to the requesting user. A calibration phase is also needed for the IPS to dynamically update its fingerprint database to deal with noisy WiFi signal measurements and physical environment changes such as indoor layout changes and additions/deletions of WiFi APs.

There is significant research on crowdsourced techniques such as [2]–[5] to reduce the calibration effort in WiFi-based IPSes. The common idea is to outsource the collection of RSS fingerprints to indoor smartphone users. These techniques combine the IMU sensor data and RSS fingerprints in various ways and demonstrate significant success. The security issue, however, is overlooked in existing studies. In particular, a crowdsourcing worker can submit fake data to induce a low-fidelity fingerprint database at the IPS for various motives. For example, the worker may want to claim rewards without actually collecting data, be hired by a malicious business competitor of the IPS operator, or misbehave just for fun.

The goal of this work is to discover the vulnerabilities of crowdsourced WiFi-based IPSes and present some countermeasures. For this purpose, we prototype a WiFi-based IPS to evaluate potential attacks and the corresponding defenses. Although this study focuses on WiFi-based IPSes, the rationals can easily extend to other types of crowdsourced IPSes. We hope that our study can promote security considerations in the early phase of designing and deploying crowdsourced IPSes.

We identify three attacks based on the information the attackers have. In the first attack, the attackers know the indoor floor plan but have no knowledge about real indoor APs. So they can generate acceptable mobility traces fitting the floor plan, which are submitted along with totally random fake RSS fingerprints. In the second attack, the attackers know both the indoor floor plan and legitimate RSS fingerprints, e.g., by walking in the indoor environment. They add noise to RSS fingerprints before submitting them. In the third attack, the attackers have the same knowledge as in the second attack. But they purposefully change the mappings between the floor plan and RSS fingerprints instead of polluting RSS fingerprints.

We also propose the corresponding defenses based on the observation that there are often some trusted indoor users such as the employees in a shopping mall. Even if the IPS operator cannot produce a high-fidelity fingerprint database based on limited trusted users alone, their data are much more trustworthy and can be used to verify the data from untrusted crowdsourcing workers. We defend against the first attack by comparing the set of APs in an untrusted submission with those in a trusted submission for the same position. To deal with the second and third attacks, we present two novel metrics to evaluate the trustworthiness of data submissions from untrusted crowdsourcing workers. The first metric considers the correlation between the RSS fingerprints for adjacent positions in the same signal trace, while the second considers the correlation between the RSS fingerprints for the same positions in different signal traces. Finally, we combine the two metrics and design an algorithm to build a high-fidelity fingerprint database resilient to malicious crowdsourcing workers.

Our contributions can be summarized as follows.

- We are the first to study the security issues in crowdsourced WiFi-based IPSes. Our principles can be easily extended to other crowdsourced IPSes.
- We present three attacks and evaluate their performance in a prototype system. We show that the attacker can induce a localization error up to 20m under the most powerful attack.

- We propose the corresponding defenses that can safeguard a crowdsourced WiFi-based IPS from malicious data injections. We experimentally show that our technique is highly resilient to the identified attacks even if the majority of crowdsourcing workers are malicious.

The rest of the paper is organized as follows. Section II introduces the background of crowdsourced WiFi-based IPSes. Section III describes the prototype system we build. Section IV presents the adversary model and experimentally evaluates three possible attacks. Section V gives our countermeasures against the identified attacks. Section VI experimentally evaluates the efficacy of our countermeasures. Section VII outlines the related work. Section VIII concludes this paper.

II. Basics of Crowdsourced WiFi-Based IPS

In this section, we introduce the basic operations of a WiFi-based IPS to help understand the proposed attacks and defenses. WiFi-based IPSes depend on the ubiquitous WiFi infrastructure in indoor environments and the penetration of WiFi-capable smartphones into people's everyday life. No additional hardware is needed to augment the network infrastructure or equip mobile users. When there is need for indoor positioning, the user turns on the IPS app on his smartphone. Assume that there are n APs in a given indoor environment, denoted by $\text{AP}_1, \ldots, \text{AP}_n$. At any specific indoor location, the smartphone can detect n RSS values (denoted by $\text{rss}_1, \ldots, \text{rss}_n$), one for each AP. If some APs are not discoverable, the corresponding RSS values are set to default system values. We refer to $\langle \text{rss}_1, \ldots, \text{rss}_n \rangle$ as an RSS fingerprint (or just fingerprint for short) for that position. Assume that the IPS operator has maintained a fingerprint database composed of the mappings between fingerprints and indoor locations. Upon receiving the user's fingerprint, the IPS operator finds the closest match in its fingerprint database and then returns the corresponding location to the user.

Radar [6] is the most classical WiFi-based indoor positioning method. It uses deterministic fingerprinting and matching based on Euclidean distance. To find the closest match in the database for a received fingerprint $\langle \text{rss}_1, \ldots, \text{rss}_n \rangle$, Radar minimizes the distance $\sqrt{(\text{rss}'_1 - \text{rss}_1)^2 + \cdots + (\text{rss}'_n - \text{rss}_n)^2}$ for an arbitrary record $\langle \text{rss}'_1, \ldots, \text{rss}'_n \rangle$ in the database.

Horus [7] improves Radar by employing probabilistic techniques to find a maximum likelihood fingerprint in the database. Given that RSS fingerprints are highly dependent on time, locations, and even devices, Horus maintains the RSS fingerprint distribution at every position x_i as

$$P(\langle \text{rss}_1, \ldots, \text{rss}_n \rangle | x = x_i) = \prod_{k=1}^{n} P(\text{rss}_k | x = x_i) \ .$$

Based on Bayesian inference, Horus derives the maximum likelihood location for each received fingerprint.

Traditional WiFi-based IPSes face two key challenges. First, it is very time-consuming and labor-intensive for the IPS operator to record the RSS fingerprints at every position in such a large indoor environment as a shopping mall. Second, dynamic calibration is needed to update the fingerprint database to deal with noisy WiFi signals, physical environment changes, and AP additions/deletions.

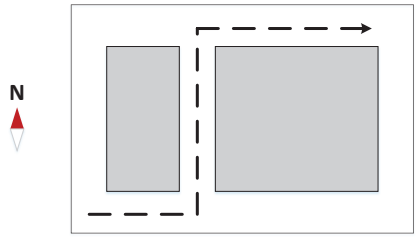

Fig. 1. Only one trace exists in the floor plan.

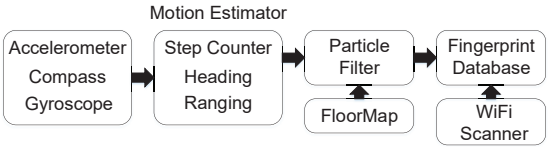

Fig. 2. The architecture of our prototype system.

III. A Prototype for Crowdsoured WiFi-Based IPS

Mobile crowdsourcing has great potential to facilitate the construction of fingerprint databases in WiFi-based IPSes. The key idea is to leverage the sensors (e.g., accelerometer, gyroscope, and magnetometer) embedded in today's smartphones in a crowdsourcing manner to automatically construct/update the fingerprint database. In such a system, a crowdsourcing worker submits the RSS fingerprints and corresponding IMU sensor data at unknown locations to the IPS operator. Different techniques such as [2]–[5] have been proposed to explore the IMU sensor data to map the fingerprints collected at unknown locations to the correct locations.

We build a prototype system based on Zee [2], a representative crowdsourced WiFi-based IPS, to illustrate the proposed attacks and defenses. Our attacks and defenses can easily extend to other systems after simple adaptations. The prototype system explores the fact that the indoor layout imposes constraints on the human mobility. For example, the user has to walk along the corridor and cannot walk through the walls or other barriers. When the user's mobility trace is known (e.g., a zigzag path), we may find only one pathway in the floor plan to accommodate the mobility trace. The more the user walks, the higher probability a single possible pathway exists. As illustrated in Fig. 1, the user walks east, turns left, turns right, and walks to the end. There is only one pathway which can accommodate such a mobility trace. If the user's mobility trace can be mapped uniquely to the floor plan, the fingerprints collected when the user walks can also be mapped to the floor plan based on timestamps.

Fig. 2 illustrates the architecture of our prototype system. The system first estimates the user motion (mobility trace) from accelerometer, compass, and gyroscope data. Then the particle filter explores the mobility trace to eliminate the particles that violate the floor constraints (e.g., it is impossible for

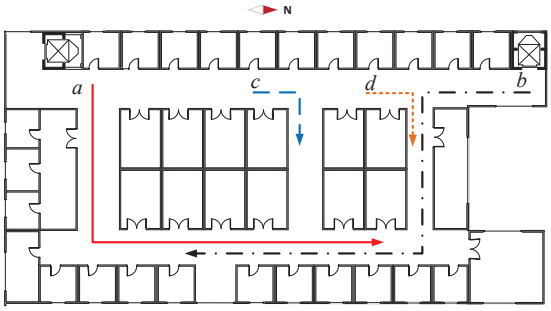

Fig. 3. Indoor floor plan for the experiment.

the user to walk through the wall). Finally, the system uniquely maps the mobility trace and associated RSS fingerprints to the floor plan. The fingerprint database is built and dynamically updated in this way. In the operating phase, the prototype system uses Horus [7] for fingerprint matching.

We implement our prototype in Google Nexus 6 based on Java. The Google Nexus 6 phone has a Quad-core 2.7 GHz Krait 450 CPU, 3 GB RAM, a 5.96-inch display, and four relevant IMU sensors (magnetometer, compass, accelerometer, and gyroscope). The sampling frequency for IMU sensors and the WiFi module are about 16.7 Hz and 0.67 Hz, respectively. We deploy the prototype on a rectangular 135m-by-35m floor of a university building with the floor plan shown in Fig. 3.

To validate the fidelity of our prototype system, we let two trusted users walk 20 times to form the initial fingerprint database with walking traces ranging between 96m to 110m long. Each user walks arbitrarily in the floor plan while carrying the smartphone in any way and orientation. In some cases, users prefer to carry the smartphone in the pocket for convenience. The sensor and RSS data are collected in two weeks to capture the RSS variations due to time. Based on the initial database, we emulate location queries at 11 random positions in the floor plan and get an average error of 2.48m and a median error of 1.72m. These results are quite consistent with the results in Zee [2].

IV. Adversary Model and Attacks

In this section, we first outline the adversary model. Then we present three attacks and report their efficacy with experimental results in our prototype system.

A. Adversary Model

Crowdsourced WiFi-based IPSes depend on the voluntary participation of many mobile users that can be recruited in various channels such as Amazon Mechanical Turk. Each crowdsourcing worker submits timestamped RSS fingerprints and concurrent IMU sensor data via an app from the IPS operator, which can be part of the actual IPS app. The IPS operator may be offering indoor positioning services for many indoor venues. In this case, it can easily associate crowdsourced data with the correct indoor venue, e.g., by checking the GPS location of the crowdsourcing worker before he enters the indoor venue. Crowdsourcing workers normally receive some rewards for their participation.

Crowdsourcing workers can misbehave for various reasons. For example, he may submit fake sensor/RSS data to claim rewards without performing the actual WiFi sensing which can be time-consuming or quickly drain his phone battery. Or he can be hired by a malicious competitor to ruin the business of the IPS operator, and such instances are not uncommon in the business world. He may also extort the IPS operator or misbehave just for fun.

The adversary can control many crowdsourcing workers, e.g., by registering many sybil accounts, teaming up with other attackers, or compromising many smartphones via malware. We are aware of the rich literature on sybil defenses which, however, still cannot eliminate fake accounts in practice. We also assume that the adversary knows our defenses.

As the first work on securing crowdsourcing-based IPSes, this paper does not have the ambition to thwart the attacks other than fake data injection. For example, one may deploy fake APs to interrupt the IPS operations or other common attacks against mobile crowdsourcing systems. These attacks deserve serious treatment in separate work.

B. Attacks

We first consider the naive attack that the attacker has no knowledge about the indoor floor plan and submits fake IMU sensor and RSS data. The mobility trace generated from fake IMU sensor data can hardly fit the floor plan, so the system can easily detect and reject the fake IMU sensor and RSS data.

Then we consider an attacker who knows the floor plan and thus can generate IMU sensor data resulting in a valid mobility trace fitting the floor plan. So the attacker merely needs to generate fake RSS fingerprints associated with valid mobility traces. Consider the floor plan in Fig. 3. We first emulate the attacker to generate a list of mobile traces which range from 40m to 400m long and can fit the floor plan (e.g., traces a and b). The starting position of each trace is random in the floor plan. As in [2]–[5], the traces that have ambiguous fittings in the floor plan (e.g., traces c and d) are not considered. We consider the following three scenarios in which the attacker tries to generate fake RSS fingerprints in different ways.

Attack-I: the attacker does not know genuine RSS fingerprints and generates fake RSS traces purely at random.

In this attack, the attacker knows neither the true RSS fingerprints for any valid mobile trace nor available APs in the indoor venue. So the attacker submits the RSS fingerprints corresponding to fake APs for each valid mobile trace. In a large indoor environment such as the shopping mall, the APs are controlled by different parties who can add, remove, replace, or move their owned APs. The IPS operator can only learn available APs from crowdsourced RSS reports and accordingly adjust the length and format of the RSS fingerprint. Specifically, the IPS operator always maintains an ordered list of APs, appends to the list any new AP learned from crowdsourced RSS fingerprints, and also removes any AP that is not seen in the RSS fingerprints from either

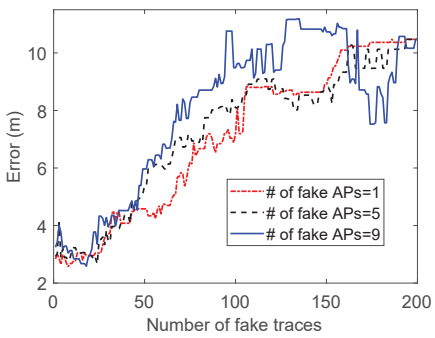

Fig. 4. Localization errors induced by Attack-I.

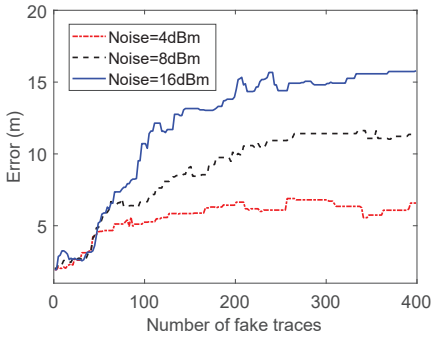

Fig. 5. Localization errors induced by Attack-II with constant noise.

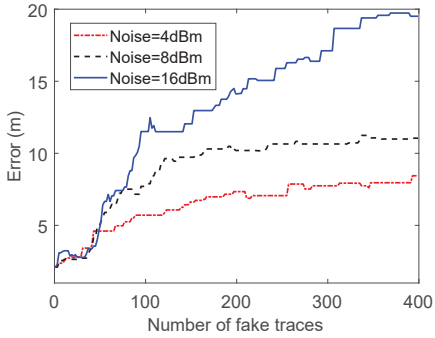

Fig. 6. Localization errors induced by Attack-II with alternating noise.

crowdsourcing workers or IPS users for a while. For example, when a new AP is discovered, the IPS operator increases the fingerprint length by one by appending a default value (say, 0) to each existing fingerprint. Therefore, the RSS fingerprints the attacker submits for fake APs will be accepted by the IPS operator and used to update the fingerprint database.

Fig. 4 illustrates the localization errors induced by Attack-I. We have about 10 genuine RSS fingerprints for each position in the database before the attack. When fake fingerprints are inserted into the database, the fingerprint distribution changes at the positions covered by fake fingerprints, so the maximum likelihood location derived by the system changes as well (see Section II). Therefore, the localization error changes dramatically with the increase of fake traces. The attacker cannot exactly control the induced errors without knowing the fingerprint database, and he can only cause random changes to the existing fingerprint distribution. As a result, we can see some temporary error fluctuations especially for the traces with more fake APs.

Attack-II: the attacker knows legitimate RSS fingerprints and adds noise to them.

In this attack, the attacker knows legitimate RSS fingerprints, e.g., by visiting the indoor venue in person or getting them from an accomplice. He then submits them after adding noise. Fig. 5 shows the localization error when a constant noise (in dBm) is added to each RSS value which ranges from -50 dBm to -90 dBm in our experiments. Again, the localization errors dramatically increase with the number of fake traces and noise strength. The errors stop quickly increasing when there are too many fake traces that start to dominate the fingerprint distribution. The attacker can also add random noise to legitimate RSS fingerprints. Fig. 6 shows a simple example, where $+r$ dBm and $-r$ dBm noises are alternately added to adjacent RSS values in a fingerprint. We can clearly see larger localization errors due to larger changes in the fingerprint distribution.

Attack-III: The attacker changes the mappings between fingerprints and indoor locations.

In this attack, the attacker knows genuine RSS fingerprints. Instead of adding noise, the attacker changes the mappings between RSS fingerprints and indoor locations. For example, let $\langle p_1, \ldots, p_\alpha \rangle$ denote the valid mobility trace that can be inferred from the attacker's IMU sensor data, where p_i ($\forall i \in [1, \alpha]$) denotes the ith position. Also assume that the genuine RSS trace corresponding to $\langle p_1, \ldots, p_\alpha \rangle$ is denoted by $\langle f_1, \ldots, f_\alpha \rangle$, where f_i is the RSS fingerprint for position p_i ($\forall i \in [1, \alpha]$). In the simplest case, the attacker maps f_i to position $p_{i+k \mod \alpha}$, where $k \in [1, \alpha - 1]$ denotes an arbitrary offset. So the attacker submits $\langle f_{\alpha-k+1}, \ldots f_\alpha, f_1, \ldots, f_{\alpha-k} \rangle$ along with his IMU sensor data (or equivalently the mobility trace $\langle p_1, \ldots, p_\alpha \rangle$) to the IPS operator. Fig. 7 shows that the average localization error under Attack-III increases dramatically with the number of fake traces for three position offsets. An IPS normally has discretized locations with constant distance (2m in our experiments) between adjacent positions. The integer-valued offset k thus has been translated into the corresponding physical distance in Fig. 7.

Attack-III is much more powerful than the previous two attacks due to its more organized nature, as it purposefully misleads existing fingerprints towards the offset direction. As we can see in Fig. 7, the attacker just needs to inject about 40 fake traces to attain the maximum localization error achievable under Attack 1 or Attack 2. In addition, Fig. 8 shows that the localization error increases linearly with the offset for a fixed number of traces, which is quite expected.

Summary of Attacks: The three attacks above are all in their basic forms. The adversary can conceive many variations of each attack or an arbitrary combination of the three attacks to disrupt the IPS operations. There is thus a pressing need to develop sound defenses to safeguard crowdsourced WiFi-based IPSes from these attacks.

V. Defenses

In this section, we present some effective defenses against the three attacks reported above. Our defenses depend on the observation that there are always some trusted users in many indoor environments (e.g., the employees in a shopping mall) who can act as trustworthy crowdsourcing workers for the IPS operator. Even though these trusted users are far from enough to build a high-fidelity fingerprint database, their data can be explored to infer the trustworthiness of RSS fingerprints submitted by unknown crowdsourcing workers.

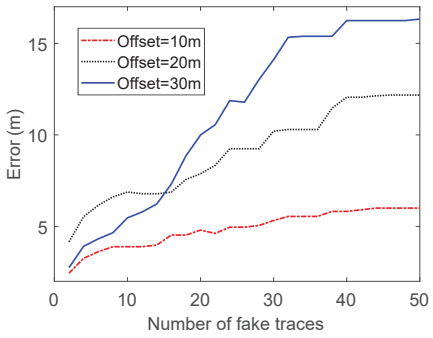

Fig. 7. Localization errors induced by Attack-III.

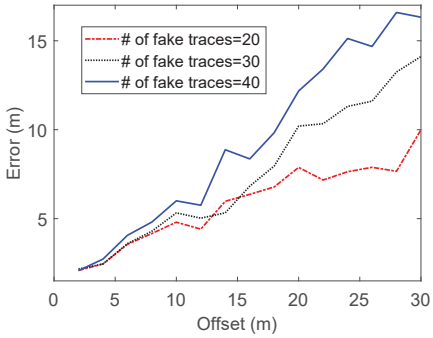

Fig. 8. Localization errors induced by Attack-III.

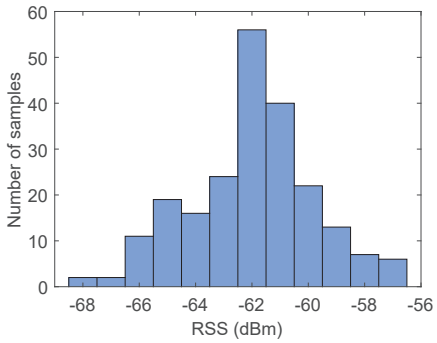

Fig. 9. Histogram of RSS values collected from an AP when the user is static for about 5 minutes.

A naive idea is to directly compare the fingerprints from unknown workers with those from trusted users for the same positions. However, wireless signal variations make such direct comparisons unreliable. For example, Fig. 9 shows the histogram of RSS values collected from an AP when the user is static for about 5 minutes. The histogram covers a large range of 11 dBm. In the crowdsourcing scenario, the histogram range can be even larger because of user mobility, inaccurate mapping from fingerprints to positions, and so on. An effective defense thus must tolerate RSS variations.

In what follows, we first present a simple method in data preprocessing to defend against Attack-I. Then we present two metrics to evaluate the trustworthiness of RSS traces from crowdsourcing workers. Finally, we present an iterative algorithm based on the two metrics to build a high-fidelity fingerprint database even in the presence of attacks. Throughout the discussion, we consider a candidate RSS trace $\langle f_1, \ldots, f_n \rangle$, where $f_i = (x_{i1}, x_{i2}, ..., x_{im})$ is the fingerprint at position i ($\forall i \in [1, n]$), and x_{ij} is the RSS value from AP$_j$ ($\forall j \in [1, m]$) collected at position i.

A. Preprocessing against Attack-I

The AP set sensed by different users in the same position should not differ too much in a short time window, or most IPSes would not work. To defend against Attack-I, we compare the APs detected by an unknown worker and a trusted user in the same position. For example, consider a worker who submitted an RSS trace $\langle f_1, \ldots, f_n \rangle$ for n positions. Let A_i and $A_i^{'}$ denote the APs detected by the trusted user and the worker at position i, respectively, for which the detection-time difference is smaller than a system threshold (say, 24 hours). The IPS operator computes the average intersection ratio

$$\delta = \frac{1}{n} \sum_{i=1}^{n} \frac{|A_i \bigcap A_i^{'}|}{|A_i|} \ .$$

If δ is larger than a system threshold, the trace $\langle f_1, \ldots, f_n \rangle$ is temporarily considered trustworthy for further processing.

B. Metric 1: Temporal Correlation within an RSS Trace

We also observe that fingerprints collected by different users tend to exhibit a similar RSS trend. For example, when the user walks towards an AP, the RSS increases gradually; when the user walks away from the AP, the RSS decreases gradually. Fig. 10 exemplifies this observation with the RSS trends collected by five different users when passing by the same AP in our prototype system. A fake trace with totally random RSS fingerprints will not be consistent with genuine traces related to the same AP. In other words, the attacker will be forced to generate fake traces with RSS trends similar to those of genuine traces.

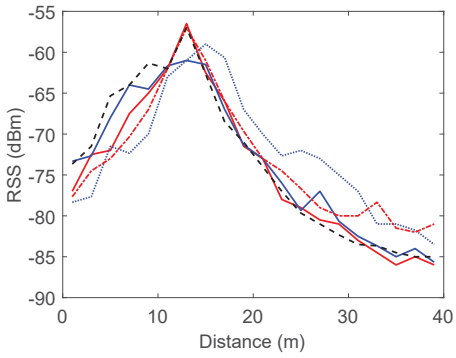

Fig. 10. RSS values collected by five users when passing the same AP.

Based on this observation, we design a temporal likelihood metric to evaluate the temporal correlation between trusted traces and crowdsourced traces. Consider a candidate RSS trace $\langle f_1, \ldots, f_n \rangle$, where $f_i = (x_{i1}, x_{i2}, ..., x_{im})$. When deriving the likelihood of observing the whole trace, we should take into account the temporal correlations among adjacent RSS fingerprints in the trace. For example, when x_{ij} is known, the range of $x_{(i+1)j}$ is largely determined because of the RSS trends illustrated in Fig. 10.

As a result, we can calculate the likelihood of observing $x_{(i+1)j}$ from AP$_j$ as

$$\mathcal{L}(x_{(i+1)j}) = \mathcal{L}(x_{(i+1)j}|x_{ij})\mathcal{L}(x_{ij}) \ ,$$

where $\mathcal{L}(x_{(i+1)j}|x_{ij})$ is the likelihood of observing $x_{(i+1)j}$ given that x_{ij} is observed from AP$_j$. We can extract a distribution for adjacent RSS variations from trusted traces and then easily compute $\mathcal{L}(x_{(i+1)j}|x_{ij})$. It is also fairly easy to calculate $\mathcal{L}(x_{1j})$ based on the distribution extracted from the

RSS values in trusted traces. So we can compute the likelihood for observing a sequence of RSS values $(x_{1j}, x_{2j}, \ldots, x_{nj})$ from AP_j as

$$\mathcal{L}(x_{1j}, x_{2j}, \ldots, x_{nj}) = \mathcal{L}(x_{1j})\mathcal{L}(x_{2j}|x_{1j}) \ldots \mathcal{L}(x_{nj}|x_{(n-1)j}) \ .$$

The temporal likelihood for the whole trace can then be computed as

$$\mathcal{L}_{\text{temporal}}(\langle f_1, \ldots, f_n \rangle) = \prod_{j=1}^{m} \mathcal{L}(x_{1j}, x_{2j}, \ldots, x_{nj}),$$

which is normalized as $\frac{1}{n}\sum_{j=1}^{m} \log \mathcal{L}(x_{1j}, x_{2j}, \ldots, x_{nj})$.

C. Metric 2: Spatial Correlation with Other Traces

Although the RSS data collected by different users may differ because of many reasons such as channel variations, phone orientation, and phone model, they generally follow a Gaussian Distribution which is exemplified in Fig. 9. We can use the distribution formed by trusted users to infer the likelihood of the RSS submitted by a crowdsourcing worker. The second metric (called *spatial likelihood*) is designed to capture the spatial RSS correlation between the fingerprints from the same position in different traces. According to [7], the RSS values from different APs collected in the same position are independent from each other. So we can estimate the likelihood of the fingerprint in position i as

$$\mathcal{L}(f_i) = \mathcal{L}(x_{i1}, x_{i2}, ..., x_{im}) = \prod_{j=1}^{m} \mathcal{L}(x_{ij}),$$

where $\mathcal{L}(x_{ij})$ refers to the likelihood of observing the RSS value x_{ij} from AP_j at position i. The spatial likelihood of the trace $\langle f_1, \ldots, f_n \rangle$ is represented as the product of likelihood in every position as

$$\mathcal{L}_{\text{spatial}}(\langle f_1, \ldots, f_n \rangle) = \prod_{i=1}^{n} \mathcal{L}(f_i) = \prod_{i=1}^{n}\prod_{j=1}^{m} \mathcal{L}(x_{ij}).$$

After normalization for different trace lengths, we can rewrite

$$\mathcal{L}_{\text{spatial}}(\langle f_1, \ldots, f_n \rangle) = \frac{1}{n}\sum_{i=1}^{n}\sum_{j=1}^{m} \log \mathcal{L}(x_{ij}) \ .$$

D. Iterative Fingerprint-Database Construction

It is well known that the RSS fingerprint database in a WiFi-based IPS needs to be periodically calibrated to deal with wireless channel variations, indoor layout changes, AP changes, and many dynamic factors in a large, complex indoor environment [2], [3]. So we present an iterative algorithm to build and maintain the RSS fingerprint database. In each updating interval (say, daily or weekly or biweekly), the IPS server always accepts new RSS traces from the trusted users first and then uses them to evaluate the trustworthiness of the RSS traces from crowdsourcing workers.

Our algorithm treats each crowdsourcing worker with equal suspicion in each updating interval. Specifically, a crowdsourcing worker may exhibit dynamic behavior by alternating between "good" and "bad" states. Reputation systems are traditional defenses against such dynamic behavior, but there are also well-documented attacks on reputation systems. For lack of space, we leave the integration of a sound reputation system in our algorithm to future work. Our algorithm applies equally to each received RSS trace without considering the past behavior of the crowdsourcing worker.

Algorithm 1: Iterative Fingerprint-Database Construction

input : Fingerprint database $\mathcal{F}$, traces $\mathcal{T}$ submitted by trusted users, traces $\mathcal{U}$ submitted by crowdsourcing workers.
output: Updated fingerprint database $\mathcal{F}$.

```
Fit all the traces in T to the floor plan and add the corresponding fingerprints to F;
if No fingerprint distribution has major change then
    return F;
Calculate RSS distribution N_i and the set A_i of APs for every position i in the floor plan;
foreach trace U in U do
    if U does not fit the floor plan then
        U ⟵ U \ {U};
    else
        Calculate A_i' for every position i in trace U;
        r ⟵ (1/n) Σ_{i=1}^{n} |A_i ⋂ A_i'| / |A_i|;
        l_U ⟵ α L_temporal(U) + (1 − α) L_spatial(U);
        if r < θ or l_U < η then
            U ⟵ U \ {U};
Rank all the traces in U according to l_U;
Add the K most trustworthy traces to F;
return F;
```

Our algorithm is designed to work even if the majority of crowdsourcing workers in a given updating interval are malicious. In particular, we rank each crowdsourced RSS trace with the two metrics above, and higher ranks indicate more trustworthiness. Only the traces with sufficient trustworthiness are added and used to update the database.

Algorithm 1 summarizes the main steps the IPS server takes in each updating interval. The IPS server first adds the traces from trusted users in the current updating interval and checks if these new trusted traces indicate any major change in the indoor environment. Specifically, the IPS server updates the fingerprint distribution for each AP at every position and compares it with the previous distribution. If the mean of any fingerprint distribution changes more than ε_1 or its variance changes more than ε_2, we consider that the distribution has significantly changed since last updating interval, where ε_1 and ε_2 are two system parameters. If none of the fingerprint distribution has changed, then there is no need to add additional unknown traces.

If the IPS server determines that there has been any sig-

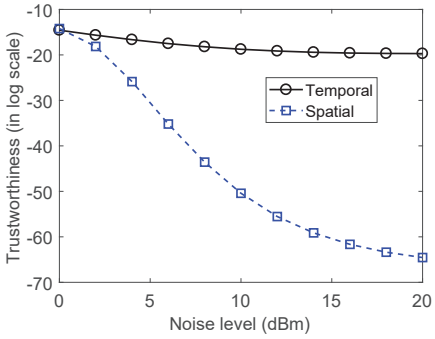

Fig. 11. Temporal and spatial trustworthiness of fake traces under Attack-II with equal noise.

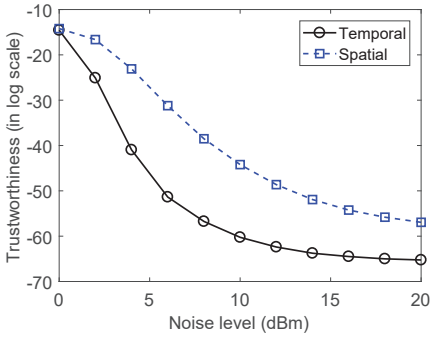

Fig. 12. Temporal and spatial trustworthiness of fake traces under Attack-II with alternating noise.

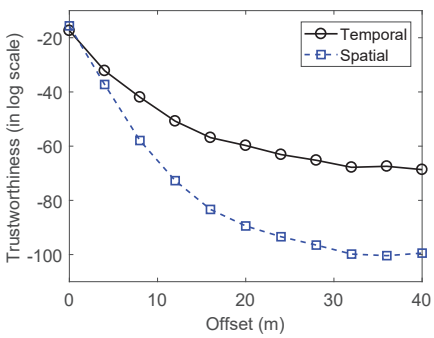

Fig. 13. Temporal and spatial trustworthiness of fake traces under Attack-III vs. offset

nificant change to any fingerprint distribution from the last updating interval, it selects more trustworthy traces to add to the database. The traces which cannot fit the floor plan will be discarded before the likelihood (trustworthiness) evaluation. Then the IPS server discards the traces subject to Attack I by checking whether the traces contain sufficiently common APs to those of trusted traces. Next, the IPS server evaluates the trustworthiness of each remaining trace by combining its temporal and spatial likelihoods in a weighted fashion. Finally, the IPS server discards all the traces with combined trustworthiness (i.e., l_U) lower than η and uses the remaining top-K trustworthy traces to update the database. The impacts of system parameters such as η and K are evaluated in Section VI.

Alternatively, the IPS operator can iteratively integrate the remaining traces into the database in the descending order of their trustworthiness until the database quality is sufficient or all the traces are used up. We ignore this option in this paper for lack of space.

VI. Countermeasure Evaluation

In this section, we report the experimental performance of our countermeasures in the prototype system. In our experiments, no fake trace generated under Attack-I passes the AP-correlation test in trace preprocessing. So we focus on the resilience of our countermeasures against Attack-II and Attack-III in this section. The floor plan for all the experiments remains the same as Fig. 3.

A. Evaluation of Metrics

In this subsection, we evaluate the two metrics on Attack-II and Attack-III, respectively. The evaluation is based on the database we build in Section III which contains 20 walking traces from trusted users. The attacker walks in the floor plan for 200m and then generates all kinds of fake traces from the learned legitimate trace.

Resilience to Attack-II. We first evaluate the performance of the temporal and spatial trustworthiness metrics under Attack-II. Fig. 11 shows the temporal and spatial trustworthiness varying with the amount of the noise added to each RSS value. As we can see, the temporal trustworthiness is relatively insensitive to the change in the amount of the noise added. This is because adding equal amount of noise to every RSS does not change the RSS trend across adjacent fingerprints for the same APs. For example, if a sequence of RSS values for the same AP in a genuine trace exhibit an ascending trend, we can still observe the same trend after the same amount of noise is added to each RSS value. In contrast, the spatial trustworthiness decreases with the increase in the noise added, as larger noise induces larger deviations of the fake fingerprints from genuine ones, and vice versa.

Fig. 12 shows the temporal and spatial trustworthiness when the attacker adds $+r$ dBm and $-r$ dBm noise alternately to adjacent RSS values in a fingerprint. As we can see, both temporal and spatial trustworthiness decrease as the amount of noise added increases, which is expected. In addition, the temporal trustworthiness decreases more rapidly than spatial trustworthiness as the amount of noise increases because the fake trace exhibits a very different temporal pattern (trend) from trusted traces considering the alternating noise added to adjacent RSS values.

Resilience to Attack-III. Fig. 13 compares the temporal and spatial trustworthiness of fake traces generated under Attack-III, where the attacker introduces different position offsets. As we can see, both metrics can provide good discrimination between fake and legitimate traces. Attack III induces dramatic differences between the fake and legitimate traces for the same positions. In contrast, the temporal trend in a legitimate trace can still be preserved to some extent in a fake trace. So we can see that the spatial trustworthiness metric outperforms the temporal one.

B. Evaluation of Fingerprint-Database Updating Algorithm

We now evaluate our fingerprint-database updating algorithm when both temporal and spatial trustworthiness metrics are employed under the following settings. In the initial phase, the database only contains four traces submitted by trusted users. The IPS server receives two legitimate traces and 40 fake traces in each updating period, as well as one trusted trace every two updating periods. The algorithm is executed in each updating period, in which the IPS server relies on the trusted traces to evaluate the trustworthiness of each unknown trace

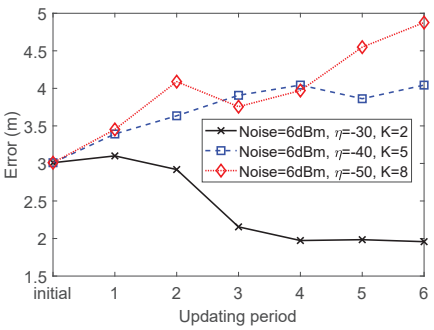

Fig. 14. Average localization error under Attack-II with 6 dBm equal noise.

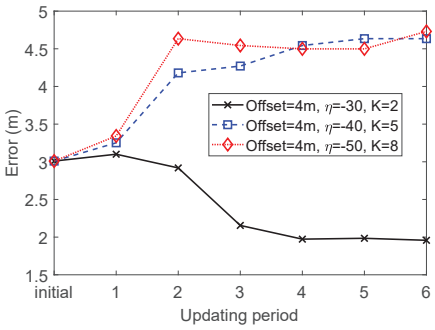

Fig. 15. Average localization error under Attack-III with 4m offset.

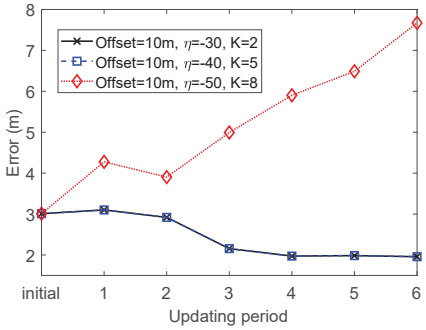

Fig. 16. Average localization error under Attack-III with 10m offset.

to distinguish legitimate traces from fake ones. Experiments for all parameter scenarios below use the same list of (trusted, legitimate, and fake) walking traces which we generate in the same way we talked in Section III.

In our experiments, we set $\theta = 0.7$, corresponding to the assumption that the APs detected by legitimate users cannot differ by more than 70% in a short time window. We also set $\alpha = 0.4$, indicating the better overall performance of spatial trustworthiness over temporal trustworthiness. In addition, we set $\eta = -30$ (log scale), because the trustworthiness (likelihood) of over 90% legitimate traces is over -30. These parameters (θ, α, and η) can be learned in practice through machine learning. The IPS server only accepts the top-K trustworthy RSS traces, where K can be estimated based on the number of trusted users and indoor traffic volume. Larger K can accelerate the convergence of database construction but also increase the vulnerability to fake traces, and vice versa.

Fig. 14 shows how the average localization error changes under Attack-II when an equal noise of 6 dBm is added to every RSS value. Our algorithm updates the database based on the top-K trustworthy traces, each with a trustworthy value $\geq \eta$. In our experiments, the two legitimate traces always have higher trustworthiness than the fake traces and are always in the top-K list for different K. The remaining $K-2$ traces in the top-K list are filled with fake traces only when there are fake traces with an trustworthy value higher than η. In general, the smaller K is, the fewer fake traces ($K-2$) added to the database, the lower the localization error, and vice versa. This trend is clearly seen in Fig. 14. When we enforce a strict criterion ($\eta = -30$ and $K = 2$), no fake trace can achieve trustworthiness equal to or over η, so only the two legitimate traces are accepted; the localization error keeps decreasing as more legitimate and trusted traces are added to the database in each updating interval and then becomes stable around the performance limit of the IPS. If we relax the criterion (e.g., $\eta = -50$ and $K = 8$), more and more fake traces are accepted, leading to increasing localization errors as time goes by.

Fig. 15 shows the average distance error under Attack-III when the attacker adds an offset of 4m to the traces. When the most restricted parameters ($\eta = -30$ and $K = 2$) are used, no fake trace is accepted in any updating interval. The average distance error keeps decreasing as more legitimate and trusted traces are included until reaching the system limit. When we relax the criteria by using smaller η or larger K, more fake traces are inserted into the database, leading to the increase in the distance errors before it becomes stable around the 4m.

Fig. 16 shows the average distance error under Attack-III when the attacker increases the offset to 10m. Under such larger offset, none of the fake traces can achieve a trustworthiness value greater than -40, so all fake traces are rejected by the IPS operator. Therefore, we can see the lines corresponding to ($\eta = -40, K = 5$) and ($\eta = -30, K = 2$) overlap with each other. On the other hand, if we decrease η to -50 and increase K to 8, some fake traces will be accepted by the database, leading to gradual increase in distance errors.

VII. Related Work

This section discusses some most germane work.

Fingerprint-based indoor positioning techniques are the most popular approaches for indoor positioning. As probably the first work along this line, Radar [6] is a deterministic localization method that employs RSS for indoor localization. Horus [7] improves Radar by keeping a fingerprint distribution for every position in the floor plan and then finding a maximum likelihood match in the database. The work [8] introduces Channel Frequency Response as a new feature for localization. SurroundSense [9] introduces more indoor features (such as light and sound) in addition to RSS fingerprints. All these methods rely on labor-intensive calibration where the IPS operator has to collect and update fingerprints for every position in the floor plan.

Model-based indoor positioning techniques estimate indoor locations using statistic models. A popular approach is to build a relation between RSS and signal propagation distance based on the RF propagation model (e.g., the log-distance path loss (LDPL)) [10], [11]. Model-based methods can dramatically decrease the need for RSS measurements but at the cost of accuracy. For example, the work in [12] evaluates some self-calibrating algorithms in office environments and finds that the median errors are consistently greater than 5m. In addition to LDPL-based schemes, there are other techniques based on Angle of Arrival (AoA) [13], [14], Time of Arrival (ToA) [15], and Time Difference of Arrival (TDoA) [16].

More recently, researchers start to explore visible light for indoor positioning. Most such techniques [17]–[21] rely on customized smart LEDs which send identification beacons for localization. Although some techniques can achieve sub-meter precision [21], it incurs significant cost to retrofit current illuminating systems. LiTell [22] first enables visible light localization on unmodified existing light hardware, but the method only applies to tube lights and the camera of smartphone must be held flat.

Simultaneous Localization and Mapping (SLAM) is a technique originating from the robotics community. SLAM relies on a robot to explore the space of interest with discrete landmarks or obstacles. Based on the laser ranging and cameras in the robot, we can determine the relative locations of the landmarks, and the robot can infer its relative location. WiFi-SLAM [23] uses a Gaussian process to model the relation of WiFi signal strengths. With more sensors embedded in the smartphone, many techniques combine IMU sensor data with human movements to realize SLAM. For example, Unloc [3] uses smartphones to sense the natural landmarks in the floor plan such as elevators and stairs, which are then connected via dead reckoning. Similar to our prototype system, Zee [2] uses the indoor constraints to map crowdsourced human mobility traces to the floor plan and then generates the fingerprint database. LiFS [4] maps fingerprints by comparing the similarity between the high-dimensional fingerprint space and the stress-free floor plan. Walkie-markie [5] presents a method for generating the floor plan based on the RSS trend when the user passes the AP.

There are also some studies on the false data injection attack in other crowdsourcing systems. For example, Zhang *et al.* [24] studies false spectrum report injection attack in crowdsourcing-based spectrum sensing. More recently, the work in [25] introduces a mechanism that explores user proximity to detect false data submitted by sybil users in crowdsourced map systems. These techniques do not consider the unique features of RSS-fingerprint-based IPSes and are not applicable to the attacks identified in this paper.

VIII. Conclusion

In this paper, we presented the first systematic study about the security issues in crowdsourced WiFi-based IPSes. We presented three attacks and evaluated their performance in a prototype system. We also designed an algorithm based on novel temporal and spatial trustworthiness metrics to generate high-fidelity fingerprint databases even if most crowdsourced RSS traces are fake. Thorough experiments confirmed that our algorithm has strong resilience to the reported attacks. Both the attacks and defenses developed in this paper can be easily extended to other crowdsourced IPSes.

IX. Acknowledgement

This work was supported in part by the US National Science Foundation under grants CNS-1619251, CNS-1514381, CNS-1421999, CNS-1320906, CNS-1700032, CNS-1700039, CNS-1651954 (CAREER), and CNS-1718078.

References

[1] "Wi-fi indoor location in retail worth $2.5 billion by 2020." [Online]. Available: https://www.abiresearch.com/press/wi-fi-indoor-location-retail-worth-25-billion-2020

[2] A. Rai, K. Chintalapudi, V. Padmanabhan, and R. Sen, "Zee: Zero-effort crowdsourcing for indoor localization," in *ACM MobiCom'12*, Istanbul, Turkey, Aug. 2012.

[3] H. Wang, S. Sen, A. Elgohary, M. Farid, M. Youssef, and R. Choudhury, "No need to war-drive: Unsupervised indoor localization," in *ACM MobiSys'12*, Low Wood Bay, UK, June 2012.

[4] Z. Yang, C. Wu, and Y. Liu, "Locating in fingerprint space: wireless indoor localization with little human intervention," in *ACM MobiCom'12*, Istanbul, Turkey, Aug. 2012.

[5] G. Shen, Z. Chen, P. Zhang, T. Moscibroda, and Y. Zhang, "Walkie-markie: indoor pathway mapping made easy," in *USENIX NSDI'13*, Lombard, IL, Apr. 2013.

[6] P. Bahl and V. Padmanabhan, "Radar: An in-building rf-based user location and tracking system," in *IEEE INFOCOM'00*, Tel Aviv, Israel, Mar. 2000.

[7] M. Youssef and A. Agrawala, "The horus WLAN location determination system," in *ACM MobiSys'05*, Seattle, WA, June 2005.

[8] S. Sen, B. Radunovic, R. Choudhury, and T. Minka, "You are facing the mona lisa: spot localization using phy layer information," in *ACM MobiSys'12*, Low Wood Bay, UK, June 2012.

[9] M. Azizyan, I. Constandache, and R. Choudhury, "Surroundsense: mobile phone localization via ambience fingerprinting," in *ACM MobiCom'09*, Beijing, China, Sep. 2009.

[10] H. Lim, L. Kung, J. Hou, and H. Luo, "Zero-configuration indoor localization over ieee 802.11 wireless infrastructure," *Wireless Networks*, vol. 16, no. 2, pp. 405–420, 2010.

[11] K. Chintalapudi, A. Iyer, and V. Padmanabhan, "Indoor localization without the pain," in *ACM MobiCom'10*, Chicago, IL, Sep. 2010.

[12] D. Turner, S. Savage, and A. Snoeren, "On the empirical performance of self-calibrating wifi location systems," in *IEEE LCN'11*, Oct. 2011.

[13] J. Xiong and K. Jamieson, "Arraytrack: A fine-grained indoor location system," in *USENIX NSDI'13*, Lombard, IL, Apr. 2013.

[14] Z. Zhang, X. Zhou, W. Zhang, Y. Zhang, G. Wang, B. Zhao, and H. Zheng, "I am the antenna: accurate outdoor ap location using smartphones," in *ACM MobiCom'11*, Las Vegas, NV, Sep. 2011.

[15] M. Youssef, A. Youssef, C. Rieger, U. Shankar, and A. Agrawala, "Pinpoint: An asynchronous time-based location determination system," in *ACM MobiSys'06*, Uppsala, Sweden, June 2006.

[16] N. Priyantha, A. Chakraborty, and H. Balakrishnan, "The cricket location-support system," in *ACM MobiCom'00*, Boston, MA, Aug. 2000.

[17] B. Xie, K. Chen, G. Tan, M. Lu, Y. Liu, J. Wu, and T. He, "Lips: A light intensity–based positioning system for indoor environments," *ACM Transactions on Sensor Networks*, vol. 12, no. 4, p. 28, 2016.

[18] B. Xie, G. Tan, and T. He, "Spinlight: A high accuracy and robust light positioning system for indoor applications," in *ACM SenSys'15*, Seoul, South Korea, Nov. 2015.

[19] N. Rajagopal, P. Lazik, and A. Rowe, "Visual light landmarks for mobile devices," in *ACM/IEEE IPSN'14*, Apr. 2014.

[20] Y. Kuo, P. Pannuto, K. Hsiao, and P. Dutta, "Luxapose: Indoor positioning with mobile phones and visible light," in *ACM MobiCom'14*, Maui, Ha, Sep. 2014.

[21] Z. Yang, Z. Wang, J. Zhang, C. Huang, and Q. Zhang, "Wearables can afford: Light-weight indoor positioning with visible light," in *ACM MobiSys'15*, Paris, France, Sep. 2015.

[22] C. Zhang and X. Zhang, "Litell: Robust indoor localization using unmodified light fixtures," in *ACM MobiCom'16*, New York City, NY, Oct. 2016.

[23] B. Ferris, D. Fox, and N. Lawrence, "Wifi-slam using gaussian process latent variable models," in *IJCAI'07*, Hyderabad, India, Jan. 2007.

[24] R. Zhang, J. Zhang, Y. Zhang, and C. Zhang, "Secure crowdsourcing-based cooperative spectrum sensing," in *INFOCOM'13*, Turin, Italy, Apr. 2013.

[25] G. Wang, B. Wang, T. Wang, A. Nika, H. Zheng, and B. Zhao, "Defending against sybil devices in crowdsourced mapping services," in *ACM MobiSys'16*, Singapore, Singapore, June 2016.