Secure WiFi Fingerprinting-based Localization

Mona A. Aboelnaga
Ain Shams University
Cairo, Egypt
Email: mona_alaa@eng.asu.edu.eg

M. Watheq El-Kharashi
Ain Shams University
Cairo, Egypt
Email: watheq.elkharashi@eng.asu.edu.eg

Ashraf Salem
Mentor, a Siemens Business
Cairo, Egypt
Email: ashraf_salem@mentor.com

***Abstract*—Accurate indoor WiFi localization is attracting more attention nowadays with the wide-spread of location-based services. WiFi fingerprinting is an indoor localization technique, where the location is estimated by mapping the strength of measured WiFi signals from different access points against a previously collected database. WiFi fingerprinting is widely adopted as being natively supported by WiFi mobile devices without any additional costs. A major concern of this technique is that an attacker could easily change the strength of received signals in order to fake the location of the mobile device. Here, we develop novel algorithms to identify attacked access points and make accurate localization in the presence of attacks. We evaluate the performance of our developed algorithms using different WiFi fingerprinting datasets under different attack models. Experimental results show that the proposed algorithms are resistant to attacks and can achieve more robust location estimation than other strategies.**

***Keywords*—Indoor localization, RSSI, Security, WiFi fingerprinting**

I. Introduction

Accurate location-awareness is an important requirement for most of the Internet of Things (IoT) objects to provide needed services. There are two categories of localization techniques: indoor localization and outdoor localization. For outdoor localization, a Global Navigation Satellite System (GNSS) such as Global Positioning System (GPS) [1], can be used. However, GPS is not suitable for indoor localization as satellite signals cannot propagate through walls or roofs [2] [3].

Indoor localization approaches are divided into two categories: one-stage and two-stage approaches [4]. For one-stage approaches, localization could be carried out using the angle of arrival (AOA), the time difference of arrival (TDOA), or the phase difference of arrival (PDOA) techniques. For two-stage approaches, a database is established during an offline training phase, which is used for location estimation during the online testing phase. Two-stage approaches mostly use received signal strength intensity (RSSI) in their two phases to get the correct location of mobile objects. WiFi fingerprinting is one of the major techniques in RSSI localization. WiFi fingerprinting requires a robust RSSI database, so-called a *radio map*, which is used to train the signal strength map as well as for matching new RSSI vectors during the online phase. Typical fingerprinting requires the signal strength to be measured from all accessible access points (AP) for each reference position (RP). During testing phase, the location of a new RSSI vector is estimated by getting the closest match from the database of the RPs. Many techniques are used for localization, some are deterministic whereas others are probabilistic [5]. Deterministic techniques use deterministic statistics to estimate the testing vector location, such as the nearest neighbor algorithm [6]. Probabilistic techniques solve the localization problem by constructing the RSSI distribution, such as the joint clustering method [7]. These localization methods are not robust in dealing with outliers as they can deviate the testing vector significantly from the stored training data, leading to large localization errors [8].

Secure localization is an important subject in WiFi fingerprinting as the RSSI readings are vulnerable to malicious attacks beside the accidental environment changes. An adversary can easily prevent correct location estimation from working correctly by modifying RSSI readings. Unfortunately, due to the nature of the problem, traditional cryptographic methods, such as authentication and encryption, cannot defend such attacks. As a result, researchers try to find robust methods for secure localization. There are two categories of secure localization approaches: signal-based and model-based approaches [9]. Signal-based approaches try to detect attacked APs and then exclude them during the location estimation phase. Model-based approaches try to find robust localization schemes which are tolerant against attacks without detecting which APs are being attacked. In this study, we focus on attack detection schemes. We focus on RSSI-based attacks that alter the RSSI readings during the online operation of the system.

Researchers made many trials to achieve secure localization. Chen et al. proposed a residual weighting algorithm that reduces the non-line-of-sight errors in localization [10]. They require trying each combination of three-APs in localization, which requires heavy computation. Zang et al. observed that finding the nearest neighbor by minimizing the median of distance instead of minimizing the Euclidean distance is more robust in the presence of attacks [11]. Chen et al. used the minimum Euclidean distance in the signal space as an indicator of the presence of attacks [12]. Kushkie et al. rejected sensor measurements that have the lowest confidence scores among the estimated covariance matrix of the Fisher method [13]. Yang et al. separated estimated locations into two clusters and then chose the cluster with the smaller distance between its candidates and its centroid [14]. Similar to the work in [10], they required trying each combination of three APs in their localization technique. Li et al. made an assumption that a similar attack is applied to all RSSI readings and as a result observed that choosing the difference between the pairs of RSSI signals is more robust than using the RSSI values directly [15]. Meng et al. tried to reject outlier readings that are due to environmental changes [8]. They proposed a probabilistic fingerprinting localization method. Laoudias et al. tried to detect the DoS (denial of service) attack on some APs by using the sum of distances to the k-nearest

neighbors (KNN) as an attack indicator [16]. Fang et al. used a probabilistic inclusive disjunction model to calculate the likelihood at each RP [17]. They collected RSSI readings 200 times for each RP to build their database. Kim et al. tried to continuously update the fingerprinting data to overcome the changes in the network [18]. Li et al. tried to mitigate attacks depending on the assumption that the RSSI values sensed by different users (in the same location) should not have big differences within a short time interval. [19]. They also tried to make use of the assumption that users with the same mobility paths will have similar RSSI measurements. Based on the previously referred work, there is no well-defined structure for the WiFi fingerprinting training data where each group of researchers makes their own assumptions regarding the offline data collection phase. For example, in [19] authors requested the fingerprinting database to be augmented with the path of motion to have their algorithm working properly.

In this paper, we try to involve attacks that manipulate the RSSI readings instead of assuming that an attacked AP will be fully blocked. Fingerprinting database is not required to be continuously updated with the assumption that the data collected in the training phase is not corrupted.

The rest of this paper is organized as follows. Section 2 provides an overview of the system and describes our attack detection algorithms. Section 3 describes the experimental setup and Section 4 presents the experimental results. Section 5 concludes the work of this paper.

II. Attack Detection

In this section, we give an overview of our proposed system and explain our attack detection algorithms.

A. System Overview

The WiFi fingerprinting system consists of an indoor WiFi network of M APs. We assume a mobile device measures the RSSI of each AP, which is represented in a vector $S = [s_1, s_2, ..., s_M]$, where s_m represents the measured RSSI from the m^{th} AP. The mobile device uses a fingerprinting localization technique to make use of such collected information to estimate its location. A fingerprinting localization process could be divided into two phases. In the offline phase, for a number of N RPs, a mobile device collects the measured RSSI vector S_n together with the coordinates L_n. For each RP, the collected data has the form of $[L_n, S_n]$. The collected data is stored in a localization database (training data) represented by a matrix R. In the online phase, the mobile device estimates its location $\hat{L}$ by comparing the measured vector (testing vector) $\hat{S}$ with the matrix R.

We consider an adversary tries to prevent such localization system from working properly. We assume the adversary has a full access over a subset of (X) APs, where he/she could increase/decrease the transmitted power. In addition, the adversary could change the identity of the attacked APs (the physical addresses of these APs). As a result, the mobile device makes an incorrect measurement of $\overline{S}$ instead of $\hat{S}$ and estimates its location to be $\overline{L}$ instead of $\hat{L}$. We refer to attacked APs as attacked features inside the vector $\overline{S}$.

We propose the following algorithms in order to find attacked features to be excluded during the localization process.

B. Detection by using Exhaustive Search

In the presence of attacks, the testing vector $\overline{S}$ is expected to be dissimilar to the training data R. When using KNN for fingerprinting localization. We can detect the presence of attacks by exploiting the distance between the testing vector $\overline{S}$ and the nearest neighbors in R [16]. We can choose the combination of $(M - X)$ APs that minimizes the distance between $\overline{S}$ and the nearest training vector in R assuming that the excluded X APs are the attacked APs.

This solution suffers from long execution time as it needs $\binom{M}{X}$ iterations. The next algorithm tries to achieve the same result with a much shorter execution time by avoiding trying all $\binom{M}{X}$ iterations.

C. Detection by Excluding Top Error-Contributing Features

This algorithm is based on the observation that the distance between $\hat{S}$ and its closest training vector $\hat{S}_n$ is much smaller than the distance between $\overline{S}$ and $\hat{S}_n$. In other words, we assume that the distance between $\overline{S}$ and $\hat{S}_n$ is mainly due to the difference of the attacked features. Based on this assumption, we could directly compare the features of $\overline{S}$ with the corresponding features of each training vector S_n and exclude the X features with the greatest difference for each training vector. After excluding these X features for each training vector, the distance is calculated between $\overline{S}$ and that training vector. The training vector with the smallest distance is assumed to be the correct training vector $\hat{S}_n$ and the excluded features are assumed to be the attacked features. The algorithm is listed in Algorithm 1. We refer to this algorithm by **TECF Algorithm**.

Algorithm 1 TECF Algorithm

```
input: maximum number of attacked APs (X)
for each testing vector do
    for each training vector do
        Find the difference of each feature between the 2 vectors
        Exclude the X features with the greatest difference
        Calculate the distance between the 2 vectors after excluding these X features
    end for
    Consider the training vector with the smallest distance as the closest training vector
    Consider the excluded X APs as the attacked APs
end for
```

D. Detection by Excluding Features with Unacceptable Differences

In the previous algorithm, we exclude X APs from each testing vector $\overline{S}$, while it is not necessary that all attacked X APs are heard by the mobile device from all testing locations. In addition, the attacker could choose to attack less than X APs. As a result, we would be corrupting the data by always excluding X APs from each testing vector especially in case of only X or less APs are heard by the testing vector. In this algorithm, we estimate the actual number of attacked APs to be excluded. We use a prior knowledge of how features change

between the k-nearest neighbors within the training data and consider this change as the legitimate change threshold. If there is a change that exceeds this threshold, it is considered to be due to an attack. We use the idea of the legitimate change threshold to validate the outputs of TECF algorithm (the attacked feature list and the proposed nearest neighbor vector) by checking if all the elements of the attacked feature list (APs) are having differences with the proposed nearest neighbor, which is greater than the threshold. If any of the features fail in this validation step, we repeat the TECF algorithm while decrementing the number of attacked APs and keep repeating the same process till all attacked features pass the validation step. Details of determining the threshold and the attack detection algorithm are shown in Algorithms 2 and 3, respectively. We refer to this algorithm by **FUD Algorithm**.

Algorithm 2 Getting the threshold used in FUD Algorithm

1: **for each** training vector **do**
2: Find the distance between each feature and the k-nearest neighbor of remaining training data
3: Record the maximum value within all features
4: **end for**
5: Draw histogram of the recorded values
6: Choose a suitable threshold from the histogram

Algorithm 3 FUD Algorithm

1: **for each** testing vector **do**
2: numberOfAttackedAPs = X
3: **while** numberOfAttackedAPs !=0 **do**
4: Run TECF Algorithm given that the max number of attacked APs equals numberOfAttackedAPs
5: Find the nearest training vector and the set of attacked APs based on Algorithm 1
6: Find the difference of each attacked feature between the nearest training vector and the testing vector
7: **if** the difference associated with any attacked AP is less than or equal to the threshold **then**
8: numberOfAttackedAPs = numberOfAttackedAPs-1
9: **else**
10: Consider the set of attacked APs the same set provided by the last run of Algorithm 1
11: break;
12: **end if**
13: **end while**
14: **end for**

E. Detection by using an Analytical Inverse Model

Previous algorithms depend on statistical observations, but here we try to detect the attacked features using an analytical inverse model. Ideally, our RSSI data should follow the path loss shadowing model, as given by (1) [20], where A_{ap} is the ap-th AP transmit power. n_{ap} is the path-loss coefficient of the ap-th AP. $d_{i,ap}$ is the distance between the ap-th AP and the i-th measurement point. $\eta_{i,ap}$ is a noise factor which is ignored in our work. To be able to use the model, we need to know A_{ap}, n_{ap} and the coordinates of each AP.

$$p_{i,ap} = A_{ap} - 10n_{ap}log_{10}d_{i,ap} + \eta_{i,ap} \quad (1)$$

Here, we use the difference between the RSSI that should be received according to the model and the actual received RSSI for detecting attacked APs. We say that if we use a set of APs, where some APs are attacked so the calculated RSSI according to the model of the correct location will differ from the actual received RSSI. If this set of APs are the unattacked ones, then the difference between the two RSSI readings should be due to noise only so it will be minimum. In this algorithm, after excluding each combination of X APs (to try all combinations and then select one), we find the location using weighted KNN algorithm (WKNN) [21] (as explained in the next section), then use inverse modeling to find the RSSI of each AP of this location. We sum the difference between the received RSSI and the calculated one for each $(M - X)$ APs combination. We choose the combination of $(M - X)$ APs that minimizes the distance between the received RSSI and the calculated one assuming that the excluded X APs are the attacked APs. This solution suffers from long execution time as it needs $\binom{M}{X}$ iterations and it assumes that we receive RSSI from more than X APs and can localize any testing vector after excluding X APs having RSSI. The algorithm is explained in Algorithm 4. We refer to this algorithm by **AIM Algorithm**.

Algorithm 4 AIM Algorithm

1: **input**: maximum number of attacked APs (X)
2: **for each** testing vector **do**
3: **for** counter from 1 to $\binom{M}{X}$ **do**
4: Exclude new combination of X APs
5: Find the location using $M - X$ APs by WKNN method
6: From the model in (1) find the RSSI that should be received at the calculated location
7: Store the difference between the calculated RSSI and the received one
8: **end for**
9: Find the minimum difference from the $\binom{M}{X}$ iterations
10: Consider the excluded X APs as the attacked APs
11: **end for**

III. Experimental Setup

To evaluate the effectiveness of our algorithms, we used two online datasets on WiFi fingerprinting with different number of APs. We refer to the first dataset as $dataset1$ [22], [21]. Its area is approximately $860m^2$. Each location in it is characterized by RSSI readings from 65 APs. The coordinates of each location are given as latitude and longitude. The intensity values are in negative integer values ranging from -99 dBm (very poor signal) to about -33 dBm. We use this training data without averaging (each location is repeated four times according to the orientation). We refer to the second dataset as $dataset2$ [23], [20]. It has a surface of about $9000m^2$. It is for a 4-floor university building. It is characterized by RSSI readings from 309 APs. The intensity values are in negative integer values ranging from -101 dBm (unheard signal) to about -20 dBm. It is clear that $dataset1$ is much smaller than $dataset2$ in terms of the number of APs and covered area.

We apply two attack models over the two datasets, which are the swapping attack and the transformation attack. In the swapping attack, we replace the IDs (MAC addresses) of

each attacked AP by another one. This method can be easily implemented by attacking the physical IDs of the APs. In the transformation attack, we add a linear attack model. The model modifies RSSI into $\alpha * (unattackedValue) + \beta$, where α and β are two attacking parameters. By setting α to -1 and β to *2*mean(minRSS,maxRSS)*, we rotate the RSSI readings around the mean value of received RSSI reading. This formula converts weak signals into strong ones and vice versa. This method is a type of a man-in-the-middle attack.

We use WKNN for localization, as explained in [21]. First, we find the k-nearest RPs to the new observed vector $\hat{S}$. Euclidean distance is used to measure the distance between each RP and the vector $\hat{S}$, as shown in (2). Then, we calculate the estimated coordinate separately $\hat{L}$, as shown in (3), where w_j is the inverse of the Euclidean distance between each k neighbor of the radio map and the new vector $\hat{S}$ ($w_j = d(s_j, \hat{s})^{-1}$). Parameter k is set to 2. We made a simple modification on the algorithm by only considering the heard APs of the testing vector as it gives a better accuracy.

$$D_i = \sqrt{\sum_{j=1}^{m} (s_{ij} - \hat{s}_j)^2} \tag{2}$$

$$\hat{L} = \frac{\sum_{j=1}^{k} w_j L_j}{\sum_{l=1}^{k} w_l} \tag{3}$$

We implemented our algorithms using MATLAB. We use the cumulative distribution function (CDF) of the localization error as a measure of accuracy. We consider an attack that affects around 10% of used APs. We neglect the APs that are rarely seen in the training data as they can be considered outliers. For $dataset1$, we neglect the APs that appear less than 10 times in R and for $dataset2$, we neglect the APs that appear less than 50 times in R.

IV. Experimental Results

In this section, we evaluate the performance of our WiFi fingerprinting attack detection algorithms and compare their performance on our datasets. We choose to compare the performance of our algorithms with the performance of the algorithm presented in [11] (referred to this algorithm by *median-based* algorithm) as it does not require any additional data to be augmented with the fingerprinting database and it could be applied to mitigate a wide range of fingerprinting attacks.

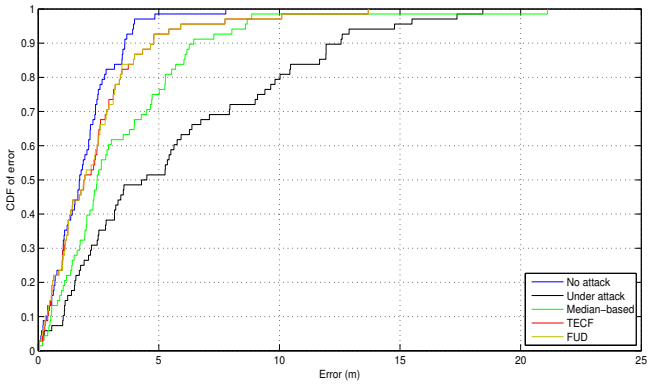
A. The Accuracy of TECF and FUD Algorithms

Table I lists the mean localization error when using our algorithms to mitigate the two attacks. Table II compares between their execution time. As shown in Table II, the exhaustive search takes around 500 ms for each testing vector, while in TECF it takes only 0.4 ms for each testing vector as a result, the TECF algorithm saves around 99.9% of the execution time. The exhaustive search has computationally failed to be applied over $dataset2$ due to the large number of APs resulting in a very large execution time.

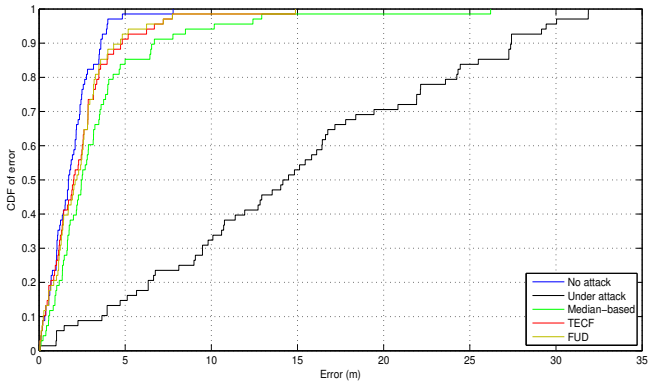
Fig. 1 and Fig. 2 show the accuracy of localizing testing vectors in the presence and the absence of attacks on both $dataset1$ and $dataset2$. Since both exhaustive search and TECF algorithms are the same algorithm but with different implementations, we do not illustrate the performance of the exhaustive algorithm in Fig. 1 and Fig. 2. We suffice with mentioning its performance in Tables I and II. Fig. 1 and Fig. 2 also show the localization accuracy of using TECF and FUD algorithms in comparison with the median-based algorithm. When applying the TECF algorithm on $dataset1$, the mean error is 2.4 in the presence of swapping attack and 2.44 in the presence of transformation attack, while for median-based the mean error is 3.41 and 3.44 for swapping and transformation attacks, respectively. Furthermore, we were able to find the attacked APs, which is not possible using the median-based approach. This could help at the correctness of data afterward. On applying the same comparison over $dataset2$, the mean error is 9.24 in the presence of transformation attack, while for median-based the mean error is 12.25. Our algorithm failed to mitigate the swapping attack since in $dataset2$, X was chosen to be 20 and we corrupted the data by always excluding 20 APs while a testing vector could be receiving data from even less than 20 APs. If we exclude the actual number of attacked APs, the algorithm shows an obvious enhancement as the mean error is 8.56, as shown in Table I. The algorithm showed good accuracy under the transformation attack as it converts each value to its inverse and hence there was always more than 20 heard APs (where all unheard APs were transformed to a heard RSSI value based on the attack formula). On applying the FUD algorithm on $dataset1$, the mean error is 2.4 in the presence of the swapping attack and 2.43 in the presence of the transformation attack while for median-based the mean error is 3.41 and 3.44 for swapping and transformation attacks, respectively. On applying the same comparison over $dataset2$, the mean error is 9.17 in the presence of swapping attack and 9.27 in the presence of transformation attack while for the median-based algorithm the mean error is 8.24 and 12.25 for swapping and transformation attacks, respectively. As a result, the FUD algorithm enhanced the localization accuracy and got rid of the limitations of the TECF algorithm.

B. The Accuracy of AIM Algorithm

As we are dealing with real fingerprinting dataset, there is a lack of information about APs and here we need to know A_{ap}, n_{ap} and the coordinates of each AP to be able to use the model. To be able to check the effectiveness of our algorithm, we simulate a new fingerprinting dataset using the path loss shadowing model. As a result the correct parameters of the model are known. We add a Gaussian noise with a mean of 2 and a variance of 5 to the generated dataset to be nearer to the real dataset. We refer to this dataset as $dataset3$. Fig. 3 shows the accuracy of localizing testing vectors in the presence and the absence of attacks on $dataset3$ and also shows the localization accuracy of using AIM algorithm in comparison with the TECF, FUD, and median-based algorithms. AIM algorithm can achieve a nearly similar accuracy as our TECF and FUD algorithms and better than the median-based algorithm by 40% in the presence of swapping attack. In the presence of transformation attack, AIM algorithm is better than median-based by 30% but gives a slight less accuracy than the other two algorithms.

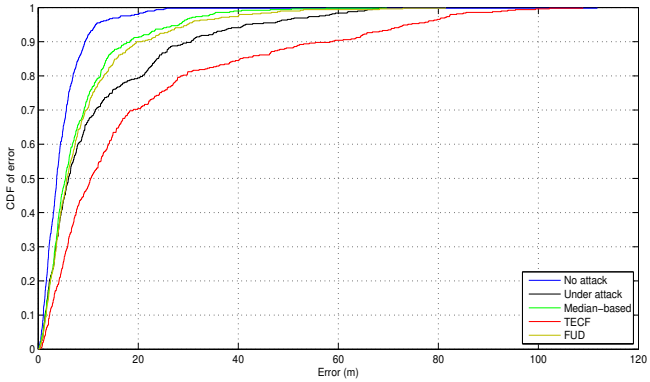


(a) Under swapping attack.

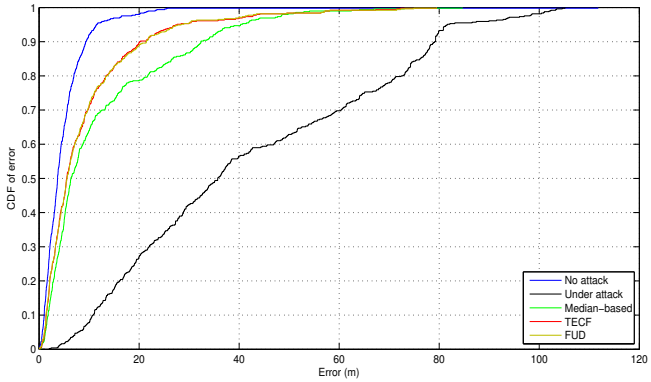


(b) Under transformation attack.

Fig. 1: Localization accuracy over *dataset1* using different attack mitigation algorithms.

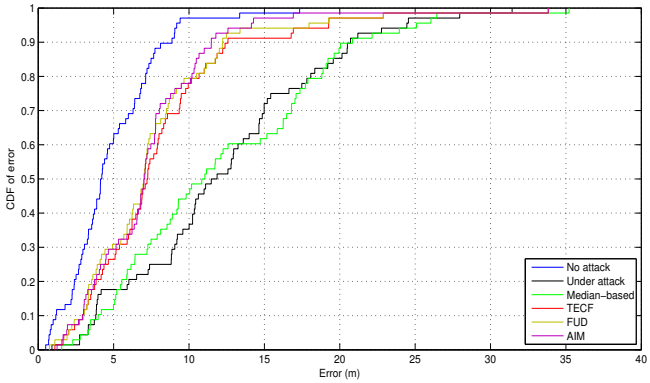


(a) Under swapping attack

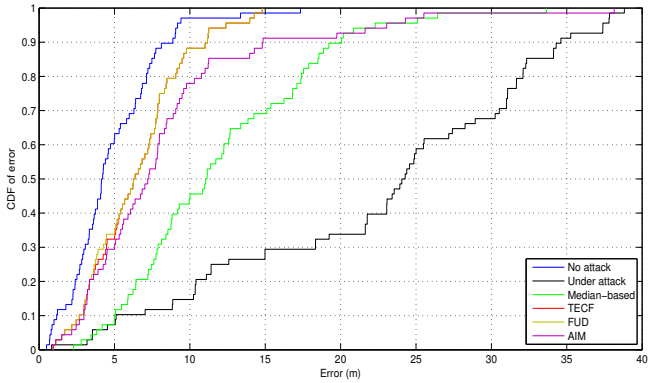


(b) Under transformation attack

Fig. 2: Localization accuracy over *dataset2* using different attack mitigation algorithms.



(a) Under swapping attack



(b) Under transformation attack

Fig. 3: Localization accuracy over *dataset3* using different attack mitigation algorithms.

TABLE I: Mean localization error after excluding attacked APs of testing dataset in meters.

	Attack	Under Attack	Exhaustive search	TECF	TECF with knowing the number of attacked APs	FUD	Median-based
Dataset1	Swapping	5.69	2.40	2.40	2.27	2.40	3.41
	Transformation	14.87	2.44	2.44	2.44	2.43	3.44
Dataset2	Swapping	11.67	N/A	19.56	8.56	9.17	8.24
	Transformation	42.01	N/A	9.24	9.24	9.27	12.25

TABLE II: Execution time of detection algorithms per testing vector in msec.

	Attack	Exhaustive search	TECF	TECF with knowing the number of attacked APs	FUD
Dataset1	Swapping	500.0	0.4	0.4	0.9
	Transformation	500.0	0.4	0.4	0.9
Dataset2	Swapping	N/A	12.0	12.0	160.0
	Transformation	N/A	12.0	12.0	25.0

V. Conclusion

Malicious attacks could corrupt the RSSI of WiFi signals received from different APs in a fingerprinting indoor localization system, which could lead to large localization errors. In this study, we proposed attack detection algorithms to detect attacked APs. We evaluated the performance of proposed algorithms under swapping and transformation attacks. The performance was evaluated based on localization accuracy and the runtime of each algorithm. Results show that proposed algorithms are able to enhance localization accuracy with the presence of attacks, while having a runtime in the order of milliseconds. As a future work, we need to apply the AIM algorithm on real datasets and enhance its execution time.

References

[1] L. Chen, B. Li, K. Zhao, C. Rizos, and Z. Zheng, "An improved algorithm to generate a Wi-Fi fingerprint database for indoor positioning," *Sensors*, vol. 13, no. 8, pp. 11 085–11 096, 2013.

[2] B. Li, J. Salter, A. G. Dempster, and C. Rizos, "Indoor positioning techniques based on wireless LAN," in *First IEEE International Conference on Wireless Broadband and Ultra Wideband Communications*, Mar 2006.

[3] B. Li, J. Zhang, A. G. Dempster, and C. Rizos, "Open source GNSS reference server for assisted-global navigation satellite systems," *Journal of Navigation*, vol. 64, no. 1, pp. 127–139, 2011.

[4] L. Chen, S. Thombre, K. Jarvinen, E. S. Lohan, A. Alen-Savikko, H. Leppakoski, M. Z. H. Bhuiyan, S. Bu-Pasha, G. N. Ferrara, S. Honkala, J. Lindqvist, L. Ruotsalainen, P. Korpisaari, and H. Kuusniemi, "Robustness, security and privacy in location-based services for future IoT: A survey," *IEEE Access*, vol. 5, no. 43, pp. 8956–8977, Apr 2017.

[5] B. Dawes and K. Chin, "A comparison of deterministic and probabilistic methods for indoor localization," *Journal of Systems and Software*, vol. 84, no. 3, pp. 442–451, Mar 2011.

[6] P. Bahl and V. N. Padmanabhan, "RADAR: An in-building RF-based user location and tracking system," in *Nineteenth Annual Joint Conference of the IEEE Computer and Communications Societies*, Mar. 26–30, 2000, pp. 775–784.

[7] M. A. Youssef, A. Agrawala, and A. U. Shankar, "WLAN location determination via clustering and probability distributions," in *Proceedings of the First IEEE International Conference on Pervasive Computing and Communications (PerCom)*, Mar. 26–26, 2003, pp. 143–150.

[8] W. Meng, W. Xiao, W. Ni, and L. Xie, "Secure and robust Wi-Fi fingerprinting indoor localization," in *International Conference on Indoor Positioning and Indoor Navigation (IPIN)*, Sep. 21–23, 2011.

[9] W. Lai, Y. Su, C. Lee, S. Fang, W. Lin, X. He, and K. Feng, "A survey of secure fingerprinting localization in wireless local area networks," in *International Conference on Machine Learning and Cybernetics (ICMLC)*, Jul. 14–17, 2013, pp. 1413–1417.

[10] P. Chen, "A non-line-of-sight error mitigation algorithm in location estimation," in *IEEE Wireless Communications and Networking Conference*, Sep. 21–24, 1999, pp. 316–320.

[11] Z. Li, W. Trappe, Y. Zhang, and B. Nath, "Robust statistical methods for securing wireless localization in sensor networks," in *Proceedings of the 4th international symposium on Information processing in sensor networks*, Apr. 24–27, 2005, pp. 91–98.

[12] Y. Chen, W. Trappe, and R. P. Martin, "Attack detection in wireless localization," in *26th IEEE International Conference on Computer Communications*, May 6–12, 2007, pp. 1964–1972.

[13] A. Kushki, K. N. Plataniotis, and A. N. Venetsanopoulos, "Sensor selection for mitigation of RSS-based attacks in wireless local area network positioning," in *IEEE International Conference on Acoustics, Speech and Signal Processing*, Mar. 31–Apr. 4, 2008, pp. 2065–2068.

[14] J. Yang, Y. Chen, V. B. Lawrence, and V. Swaminathan, "Robust wireless localization to attacks on access points," in *IEEE Sarnoff Symposium*, Mar. 30–Apr. 1, 2009.

[15] X. Li, Y. Chen, J. Yang, and X. Zheng, "Designing localization algorithms robust to signal strength attacks," in *Proceedings IEEE INFOCOM*, Apr. 10–15, 2011, pp. 341–345.

[16] C. Laoudias, M. P. Michaelides, and C. G. Panayiotou, "Fault detection and mitigation in WLAN RSS fingerprint-based positioning," *Journal of Location Based Services*, vol. 6, no. 2, pp. 101–116, 2012.

[17] S. Fang, C. Chuang, and C. Wang, "Attack-resistant wireless localization using an inclusive disjunction model," *IEEE Transactions on Communications*, vol. 60, no. 5, pp. 1209–1214, Apr 2012.

[18] Y. Kim, H. Shin, Y. Chon, and H. Cha, "Crowdsensing-based Wi-Fi radio map management using a lightweight site survey," *Computer Communications*, vol. 60, no. 7, pp. 86–96, Apr 2015.

[19] T. Li, Y. Chen, R. Zhang, Y. Zhang, and T. Hedgpeth, "Secure crowdsourced indoor positioning systems," in *Proceedings IEEE INFOCOM*, Apr. 15–19, 2018.

[20] S. Shrestha, J. Talvitie, and E. S. Lohan, "Deconvolution-based indoor localization with WLAN signals and unknown access point locations," in *International Conference on Localization and GNSS (ICL-GNSS)*, Jun. 25–27, 2013.

[21] G. Jekabsons and V. Zuravlovs, "Refining Wi-Fi based indoor positioning," in *Proceedings of 4th International Scientific Conference Applied Information and Communication Technologies (AICT)*, Apr. 22–23, 2010, pp. 87–95.

[22] http://www.cs.rtu.lv/jekabsons/, 2010, [Online; Last access on 15th-Nov.-2018].

[23] http://www.cs.tut.fi/tlt/pos/Software.htm, 2014, [Online; Last access on 15th-Nov.-2018].