2015 IEEE 12th International Conference on Mobile Ad Hoc and Sensor Systems

Efficient Privacy-Preserving Fingerprint-based Indoor Localization using Crowdsourcing

Patrick Armengol
Dept. of Computer Engineering
University of Central Florida
Orlando, Florida 32816
Email: parmengol@knights.ucf.edu

Rachelle Tobkes
Dept. of Computer Science
Florida International University
Miami, Florida 33174
Email: rtobk001@fiu.edu

Kemal Akkaya, Bekir S. Çiftler, and İsmail Güvenç
Dept. of Electrical & Computer Engineering
Florida International University
Miami, Florida 33174
Email: kakkaya—bcift001—iguvenc@fiu.edu

***Abstract*—Indoor localization has been widely studied due to the inability of GPS to function indoors. Numerous approaches have been proposed in the past and a number of these approaches are currently being used commercially. However, little attention was paid to the privacy of the users especially in the commercial products. Malicious individuals can determine a client's daily habits and activities by simply analyzing their WiFi signals and tracking information. In this paper, we implemented a privacy-preserving indoor localization scheme that is based on a fingerprinting approach to analyze the performance issues in terms of accuracy, complexity, scalability and privacy. We developed an Android app and collected a large number of data on the third floor of the FIU Engineering Center. The analysis of data provided excellent opportunities for performance improvement which have been incorporated to the privacy-preserving localization scheme.**

***Index Terms*—Efficiency, fingerprint, localization, privacy, WiFi**

I. Introduction

With the proliferation of smart wireless devices, indoor localization schemes began utilizing existing communication infrastructures, referred to as WiFi Access Points (APs), within buildings to improve localization accuracy [1]–[3]. This led to development of many different approaches based on the needs of the applications. For instance, while some of them used triangulation techniques [4] others relied on a fingerprinting approach where the schemes are first trained to build a fingerprint database [5]. Compared to triangulation approaches, these schemes provided better accuracy at the expense of training costs.

While significant research has been devoted to improve the performance of these schemes, such works ignored the privacy concerns that arose as a result of being localized and tracked within the buildings. Since smart devices such as phones are constantly being carried by users, outside parties can collect a clients scanned fingerprint and use data such as the Received Signal Strength (RSS) values and a client's MAC address to track the user. This information can be collected and analyzed for a long period of time to understand the habits of individuals. Previous studies indicate that malicious individuals have used such data to expose a user's daily activities [5].

To this end, a number of solutions have been proposed in the literature. Basically, the idea is either to hide the ID of the user or the data of the user from the APs. One of the widely suggested solutions is to disguise the MAC address by dynamically changing it [6]. The motivation behind this solution is to prevent tracking of a device, since each device has a unique MAC address that is sent to WiFi access points. Different versions of this MAC randomization has been suggested in order to make it harder to analyze the collected data for tracking. These solutions include using opportunistic MAC address changing and mixed zones to decide when and where to change the address [7]. Nonetheless, this solution is not a viable option in most cases since changing MAC address is not possible for all the smart devices due to root access restrictions.

Therefore, we target the other option of hiding the RSS data when used in conjunction with a fingerprinting-based localization approach. As mentioned, the fingerprinting-based approach relies on the training data which is typically stored on a remote server by the localization service provider. When this database is accessed, a client's location is determined by comparing their RSS values for surrounding APs to the collection of fingerprints in this database. Then, a one-dimensional Euclidean distance algorithm is used to find the k-nearest neighbors whose centroid is calculated as the location of the client. This method allows the service provider or any outside attacker to track the client easily. To remedy this privacy issue, one possible solution is to use homomorphic encryption on the server-side computations. Homomorphic encryption allows for arithmetic operations to be performed on encrypted data. In this way, the location information can be returned to the user without exposing any RSS data to the service provider [5].

In this paper, we study the performance and deployment aspects of a fingerprint-based localization application that also preserves privacy. While the privacy-preservation idea was proposed in [5], the testing of the approach was done based on several assumptions which hindered the applicability of the approach in realistic environments. To this end, we developed a new Android app which can localize users on the third floor of FIU Engineering Center. Our contributions can be listed as follows: 1) We propose two new algorithms: the Missed Constant Algorithm (MCA) and the Dynamic Matching Algorithm (DMA) which allow localization in large infrastructures with the usage of fingerprints from heterogeneous devices;

978-1-4673-9101-6/15 $31.00 © 2015 IEEE
DOI 10.1109/MASS.2015.76

2) We developed a crowdsourcing-based training application to collect the fingerprint data from volunteers in the ECE department so that the data analysis can be more meaningful; 3) We study performance trade-offs in terms of computational complexity and localization accuracy; and 4) As opposed to [5], we consider a dynamic environment where the number of APs can change in time and space.

The structure of this paper is as follows. In the next section, we summarize the related work. Section III describes the system model, Section IV describes the proposed approaches, and Section V provides a comprehensive evaluation of the application. Finally, Section VI concludes the paper.

II. Related Work

A. MAC-Address Spoofing

MAC-address spoofing or dynamic MAC-address changing is an approach for privacy preservation when it comes to indoor WiFi localization. A user's smart phone constantly probes nearby access points (APs) in order to find a 'better' AP connection [8]. This allows tracking of the client's location over time. Gruteser and Grunwald [6] discuss a MAC-address changing scheme which prevents third parties from being able to distinguish clients due to the client's unique identifier constantly changing. They propose a forward chain of MD5 hashes to ensure that each generated MAC-address is valid and unlinkable to the previous. The first 24-bits of the address is chosen from a specific list of Organizationally Unique Identifiers (OUI) which is selected via using the 3 least significant bits of the MD5 hash to index into an OUI table. The remainder of the address is then created by concatenating the next 24 bits of the hash to the OUI.

With MAC-address spoofing comes several disadvantages. Gruteser and Grunwald note that when various users randomly modify their MAC-address, the problem of collision arises. Moreover, network disruptions occur on association with an access point which leads to disconnecting of WiFi on the client's device due to the client needing to acquire a new IP address.

B. Mix Zones

Beresford and Stajano [7], [9] propose an alternative solution that is used to enhance localization privacy. Referred as the *Mix Zone* model, the representation limits the areas in which users can be located. The model divides the area into two types of zones: mix zones and application zones. Mix zones are positions where the application does not receive information about the client, while application zones are positions in which users register their location. Whenever a mix zone is entered, clients switch to a new pseudonym which prevents third parties from distinguishing clients in the mix zone. This method requires the application to retrieve the state of the client from elsewhere. Additionally, a trusted third party must be established. In this paper, we determine the location of a client by comparing his/her fingerprint to fingerprints in a database, rather than having to retrieve the saved state of the client.

C. Fingerprint-Based Localization

Similar fingerprint-based methods have been proposed which involve privacy of the user and the service provider. In [5], they discuss a privacy-preserving fingerprint localization scheme referred to as PriWFL. PriWFL is split into two phases. The first phase known as the training phase is where authorized individuals collect fingerprints at every position in a specified area. These fingerprints are stored in a database maintained by the service provider. The second phase known as the online operating phase is the calculation of the to-be localized client. This is done by the client scanning the area for APs, and finding the Euclidean distance between the results of the scan and each of the fingerprints in the database. To ensure privacy in their implementation, the authors employ the Paillier cryptosystem scheme [5].

This implementation uses a fixed number of APs and requires that all of the APs are present in every fingerprint. However, this scheme does not work for larger infrastructures due to APs only having a limited range. Moreover, since we are using a crowd-sourced database with different phone types to gather our data, each phone will produce a different result and therefore will make it very unlikely that every AP in the set is included in each fingerprint. In our paper, we discuss two new algorithms which allow localization of larger infrastructures by using a crowd-sourced database.

III. System Model

The fingerprint-based application relies on the availability of training data in a remote server that is accessible by the client's device. Therefore, we setup a database server in the ECE Department using MySQL to store training data. The connections to this server is through either WiFi or LTE and relies on JDBC for accessing the data. We also obtained a floor map of the third floor to depict fingerprint locations. The database contains a table with the following columns:

- Map ID
- Location on the map
- MAC-address of an AP
- RSS value associated with the AP
- Device manufacturer, model, and software version
- Date and time

We also developed a training app which allows Android users to train the application and store data to a remote database as in Fig. 1. In this way, we were able to involve a large number of users and heterogeneous devices for creating and storing training locations. For each training location, we did multiple scans in order to increase the accuracy. As a result, we obtained a fingerprint database as follows: $\mathcal{D} = \langle (x_i, y_i), V_i = \{\nu_{ij}\}_{j=1}^{N_{\mathrm{AP}}} \rangle_{i=1}^{N_{\mathrm{F}}}$ where (x_i, y_i) is the location of the fingerprint, V_i is the fingerprint data set where each v is a fingerprint RSS value associated with AP_i, N_{AP} is the total number of fingerprint reading scan result APs for a location, and N_{F} is the total number of fingerprints in the database.

Once the training phase is completed, the localization app can utilize this database to determine a client's location

through computations that will be detailed next. A general overview of the localization approach is provided in Fig. 2.

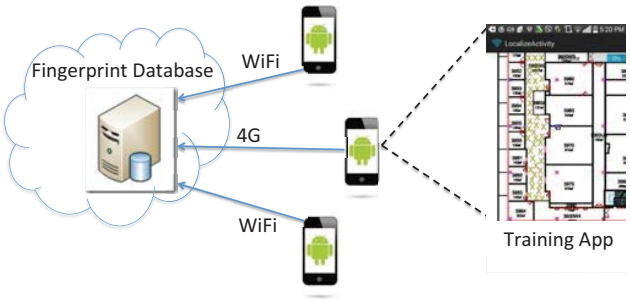

Fig. 1. Crowdsourcing-based training model.

IV. Fingerprint-based Localization Approach

In this section, we first briefly describe the background on fingerprint-based localization and then explain our proposed algorithms for the Euclidean distance computation done on the server.

A. Background

The basic process chain that is executed when a client wants to be localize itself using privacy-preserving localization is detailed in Fig. 2. The client first scans for nearby WiFi APs and their corresponding RSS values on an Android device. The scan returns a list of (AP, RSS) pairs. Calculations on the data are performed by the client in preparation for the remote request. The client then sends a request containing the prepared scan data to the service provider. The service provider retrieves relevant data from the fingerprint database and computes the Euclidean distance between each RSS fingerprint and the real-time RSS scan sample by using the technique described in [5] that considers homomorphic encryption. The resultant $(location, distance)$ pairs are sent back to the client. The pairs are decrypted at the client and sorted. At the end, the client's position is displayed as a weighted centroid for the k nearest neighbors in the sorted list.

The approach in [5] uses a distance computation algorithm which increases the computational complexity by requiring unnecessary computations on all fingerprints in the database. Additionally it assumes that there can be a set of at least N APs within range of any position within a building. When considering larger scale buildings where coverage by each AP is not absolute, the algorithm will falter during execution since N will be either too small to provide accurate results, or 0 in which case an error will occur during the computation. Although this algorithm would work well in small scale buildings, we need an alternative method to extend the privacy preserving localization scheme to all possible environments. In this section, we propose two new algorithms that implement a privacy preserving localization scheme for any scale of building while reducing the total computational complexity.

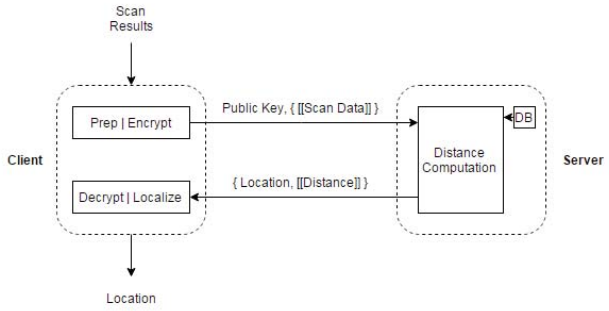

Fig. 2. Overall process for privacy-preserving fingerprint-based localization. *Note: The use of brackets [[v]] on the value v denotes that the it is encrypted.*

B. Miss Constant Algorithm (MCA)

One of the issues that is faced in fingerprinting is the number of APs that can be detected from different locations of large-scale buildings. Since the number of APs will differ significantly at any location, a certain localization or training scan may not hear from all of the APs. When a localization scan finds an AP that is not detected in a particular fingerprint, we can assume that the fingerprint is not in range of the AP and therefore compensate for that 'miss' with the use of a constant and low RSS value in the distance computation.

Based on this assumption, the proposed algorithm has three phases as detailed below:

1) Preparation Phase: A client needing to be localized will need to first do some client-side processing on the received scan data set $V' = (v'_1, v'_2, ...v'_N)$ where each v' is a scan RSS value associated with an AP, and N is the total number of localization scan results. The device calculates $S_{2\text{comp}} = \{-2v'_1, -2v'_2, ..., -2v'_N\}$ and $S_3 = \sum_{j=1}^{N} v'^2_j$. It then encrypts the prepared localization data and sends a request to the server with $[[S_{2\text{comp}}]]$, $[[S_3]]$, a list of MAC addresses for the sample scan's APs $scanAPs$, and the public key p_k to be used for encryption.

2) Distance Computation Phase: When the server receives a request by a client, it will first query the fingerprint database D and gather relevant fingerprint data. The server then calculates the squared Euclidean distance $[[d_i]]$ between the localization scan V' and each fingerprint V_i.

Using a variation on the homomorphic encryption method for calculating the Euclidean distance described in [5], $[[d_i]] = [[S_{i,1} + S_{1,2} + S_3]] = [[S_{i,1}]] \cdot [[S_{i,2}]] \cdot [[S_3]]$. Since $[[S_3]]$ is already known, all that is left to compute is $[[S_{i,1}]] = [[\sum_{j=1}^{N} v_{i,j}^2]]$ and $[[S_{i,2}]] = [[\sum_{j=1}^{N}(-2v_{i,j} \cdot v'_j)]] = \prod_{j=1}^{N}[[(-2v'_j)]]^{v_{i,j}}$.

The algorithm goes through all N localization scan results and checks if there is a match in each of the fingerprint AP lists of size N_{AP}. If the AP is contained in the fingerprint, v_i is used in the accumulation; otherwise v_i doesn't exist and we must use a constant RSS value $v_c = -120$ in its place to account for the miss. Finally when all computations are finished the server sends back $\{(x_i, y_i), [[d_i]]\}_{i=1}^{N_{\text{F}}}$.

3) Location Retrieval Phase: The client receives the response from the server containing the result list $\{(x_i, y_i), [[d_i]]\}_{i=1}^{N_\mathrm{F}}$ and decrypts the encrypted distance values. It then sorts and chooses the k smallest location-distance pairs $\{(x_i, y_i), d_i\}_{i=1}^{k}$. Finally it calculates the weighted centroid of the k points where the weight for each point is calculated with $w_i = \frac{1-(d_i/\sum_{j=1}^{k} d_j)}{k-1}$ and the location of the centroid is determined with $(c_x, c_y) = \sum_{i=1}^{k} w_i \times (x_i, y_i)$.

C. Dynamic Matching Algorithm (DMA)

Because MCA takes into account all of the localization scan results, the distance computation phase must perform calculations for each scan result. Since this phase involves arithmetic operations on homomorphic values, this computation may take up an exorbitant amount of time, especially if the device being localized returns a large resultant localization scan data set.

This algorithm aims to reduce the computational overhead in the distance computation phase while attempting to also improve the accuracy of the results. This proposed algorithm takes into consideration only the APs which are in both the localization scan data set and the fingerprint data set.

The proposed DMA algorithm has three phases as detailed below:

1) Preparation Phase: This phase will be similar to that of MCA explained before. The device calculates $S_{2\mathrm{comp}} = \{-2v'_1, -2v'_2, ..., -2v'_N\}$ and $S_{3\mathrm{comp}} = \{v'^2_1, v'^2_2, ..., v'^2_N\}$. It then encrypts the prepared localization data and sends a request to the server with $[[S_{2\mathrm{comp}}]]$, $[[S_{3\mathrm{comp}}]]$, a list of MAC addresses for the sample scan's APs $scanAPs$, and the public key p_k to be used for encryption.

2) Distance Computation Phase: When the server receives a request by a client, it will first query the fingerprint database D and gather relevant fingerprint data. The server then calculates the squared Euclidean distance $[[d_i]]$ between the localization scan V' and each fingerprint V_i.

Using a variation on the method described in the previous algorithm, $[[d_i]] = [[S_{i,1} + S_{1,2} + S_{i,3}]] = [[S_{i,1}]] \cdot [[S_{i,2}]] \cdot [[S_{i,3}]]$. The server needs to compute $[[S_{i,1}]] = [[\sum_{j=1}^{M_i} v_{i,j}^2]]$, $[[S_{i,2}]] = [[\sum_{j=1}^{M_i}(-2v_{i,j} \cdot v'_j)]] = \prod_{j=1}^{M_i}[[(-2v'_j)]]^{v_{i,j}}$, and $[[S_{i,3}]] = [[\sum_{j=1}^{M_i} v'^2_j]]$.

The algorithm goes through all N localization scan results and checks if there is a match in each of the fingerprint AP lists of size N_{AP}. If the AP is contained in the fingerprint, v_i is used in the accumulation; otherwise the AP is not considered for use in the accumulation. The total number of matches is $M_i \subseteq N$. Finally when all computations are finished the server sends back $\{(x_i, y_i), [[d_i]], M_i\}_{i=1}^{N_\mathrm{F}}$.

3) Location Retrieval Phase: The client receives the response from the server containing the result list $\{(x_i, y_i), [[d_i]], M_i\}_{i=1}^{N_\mathrm{F}}$, decrypts the encrypted distance values, and normalizes them by dividing by the number of matches M_i. It then sorts and chooses the k smallest location-distance pairs $\{(x_i, y_i), d_{ni}\}_{i=1}^{k}$.

Finally it calculates the weighted centroid of the k points where the weight for each point is calculated with $w_i = \frac{1-(d_{ni}/\sum_{j=1}^{k} d_{nj})}{k-1}$ and the location of the centroid is determined with $(c_x, c_y) = \sum_{i=1}^{k} w_i \times (x_i, y_i)$.

D. Further Performance Improvement via Filtering

The computations in the distance computation and location retrieval phases are quite costly due to the server and client having to consider all fingerprints (the count is denoted as A) in the database as possible candidates for localization. This overhead is largely undesired especially if the client intends to localize in short time or continuously in short intervals such as for indoor navigation. The mobility prediction scheme proposed in [5] improves the computational overhead of the algorithm by only considering the fingerprint locations that the client is likely to be at, based on the time and place of the previous location estimate. By doing so, the size of possible localization candidates is reduced from A to the number of fingerprints in a certain radius from the previous location. The major limitation of this technique is that it can only be used in a closely subsequent scan to the previous one. This is fitting for a navigation application where a client is continuously localizing. However, the initial localization or any localization after a relatively long period since the last would consider all A fingerprints in the database which is still costly.

To be used for general cases, we propose three new filtering techniques that can be implemented for our proposed algorithms in order to reduce their computational overhead and increase the accuracy of the results. These filtering techniques may be implemented together to decrease the computational overhead even further.

1) Fingerprint Similarity Filtering (FSF): This technique is used to discard fingerprints which are dissimilar to that of the localization scan from consideration before they are used in the intensive calculations. Given a percentage threshold $P\%$, if the list of APs associated with the fingerprint contains greater than $P\%$ of the list of APs in the localization scan, then the fingerprint is considered a viable localization candidate and is used in the algorithm. This technique drastically reduces the overhead for the distance computation phase and since the size of response list created by the server is reduced from to the number of candidates rather than the total number of fingerprints in the database N_F, it also reduces the overhead in the location retrieval phase. This is especially true for relatively slow devices that have excessively lengthy execution times for decryption.

Considering we are only dealing with fingerprints above a threshold of similarity to the scan results, the accuracy of the localization results will also improve. This provides notably more accurate and precise results for the Dynamic Matching algorithm because, unlike the the Miss Constant algorithm, it does not have a mechanism that compensates for the absence of an AP in a fingerprint.

2) Known AP Filtering (KAF): Having knowledge and keeping track of all stationary APs within a building allows for the avoidance of consideration of temporary APs such as wireless hotspots. Only the stationary APs inside the building are used. This method can be used to filter the localization scan

results received by the client in the preparation phase. Since the filtering is done early in the process, it will reduce the execution time of all three phases. This filtering method may increase the accuracy of the localization due to the exclusion of mobile or temporary APs that would appear in the scan results and throw off the distance calculations in both algorithms.

3) Device Model Filtering (DMF): Only considering fingerprints associated with the client's device greatly decreases computational overhead. Rather than retrieving all fingerprints from the database, only the ones with a specified combination of manufacturer, model, or software development kit are queried. This technique assumes that the crowd-sourced database contains sufficient fingerprints from several devices.

V. Performance Evaluation

In this section, we discuss the results of our various experiments used to test accuracy, complexity, and scalability of our algorithms and filtering methods. Our testing environment consists of an Android platform on a Qualcomm Snapdragon801 smart phone (LG G3) running at 2.5 GHz with 3 GB of RAM. We use this device to test the performance of our scheme. We run a remote server on a Lenovo T440p laptop with a 64-bit i7-4700MQ processor running at 2.4 GHz with 4 GB RAM. The fingerprints contained in the database were collected from several different devices, including: Samsung Galaxy S5, LG Nexus 5, Samsung Galaxy S4, Motorola Moto G, HTCONE, and BLU Studio 5.0.

We have considered the following metrics for assessing the performance: 1) Accuracy: indicates the success of the algorithm in localizing the client; 2) Execution time: indicates the computational time for localizing a client.

A. Ideal RSS Constant

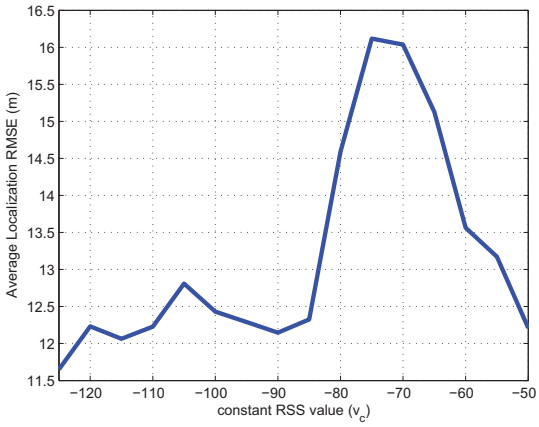

Fig. 3. Localization accuracy drops as RSS constant increases.

Determination of the ideal constant RSS value v_c is a challenge since it may affect the accuracy and performance significantly. Therefore, we conducted a detailed analysis on the collected training data using MATLAB.

The analysis revealed that the most accurate results are obtained with a constant RSS value of -125 dBm as shown in Fig. 3, i.e. Average Root Mean Square Error (RMSE) is lowest for given constant RSS value for missed APs. The average RSS value of all measurements from all the MAC addresses is -75.8 dBm. It can be seen that localization error peaks around that value. Thus avoiding the use of mean value for RSS constants improves the accuracy of localization significantly.

B. Performance under Varying FSF Percentage

We analyzed the execution time and accuracy of MCA and DMA when the FSF technique is used with varying the amount of filtering. Fig. 4 depicts the outcome of our experiments in terms of time complexity. At no filtering, the total average execution time is 43.026 sec for MCA and 17.893 sec for DMA. From 0% to 60%, DMA is shown to have a smaller computational overhead than MCA. However, when the filtering reaches 70% the execution time for DMA surpasses that of MCA. This is due to a decreased number of candidates considered and subsequently a decrease in execution time for the distance computation phase. In this case, MCA is less efficient since it needs to do more computations for the misses. It also increases execution time for the preparation phase, in which DMA is less efficient.

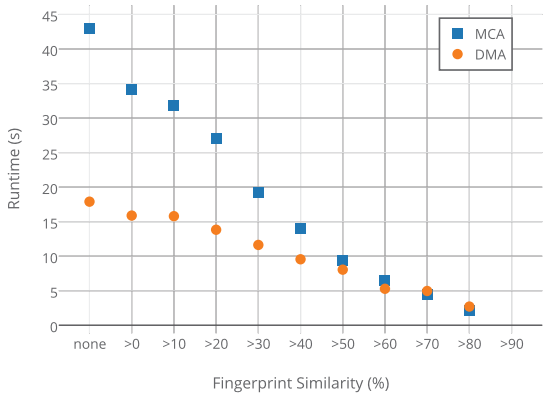

Fig. 4. Algorithm runtime using different percents of FSF.

The localization accuracy for same scenario is also analyzed as seen in Fig. 5. The average root mean square error (RMSE) is decreasing, i.e., the average localization accuracy increases with a higher degree of filtering. This indicates that higher degree of filtering helps improve accuracy and reduce computational overhead.

C. Time Complexity of Algorithm Phases

In this section, we examine the difference between our proposed algorithms, using FSF, with at least 50% and 75% thresholds. As shown in Fig. 6, during the preparation phase, MCA is more efficient. This is due to DMA having to encrypt two separate lists where MCA only needs to encrypt one. However, during the distance computation phase, DMA is much more efficient as it avoids a large amount of Paillier arithmetic that MCA is required to execute. MCA's overhead

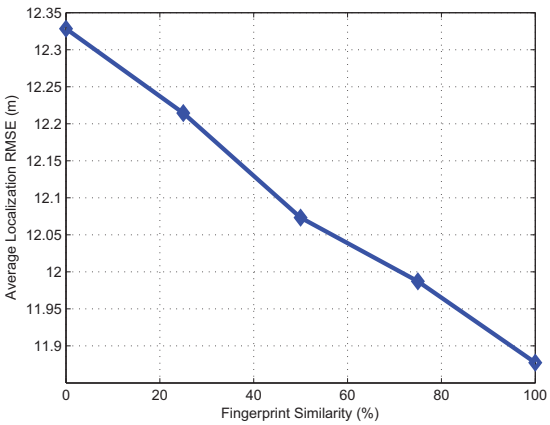

Fig. 5. Average Localization RMSE of algorithms using FSF.

for the location retrieval phase is similar to that of DMA's, trailing slightly behind as it requires an extra division operation for the entire response list. At the filtering threshold of >75%, the total execution times for both algorithms are almost equivalent.

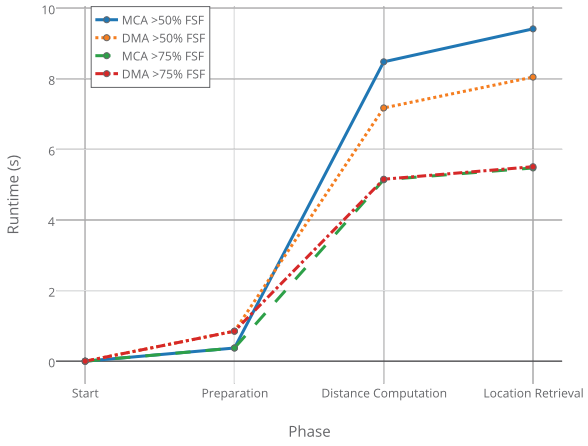

Fig. 6. Runtime for phases of algorithms using FSF.

D. Execution Times: PriWFL vs Ours

We finally compared and analyzed our two proposed algorithms using FSF with filtering greater than 75% to PriWFL. As shown in Fig. 7, both MCA and DMA's execution times were almost analogous with respect to the >75% filtering. Both of our algorithms depict an overall execution time of less than 8 seconds at the end of the retrieval phase, while PriWFL's is just under 16 seconds. Moreover, in the preparation phase alone, PriWFL's execution takes about 12 seconds while both MCA and DMA's preparation phases take about 7 seconds.

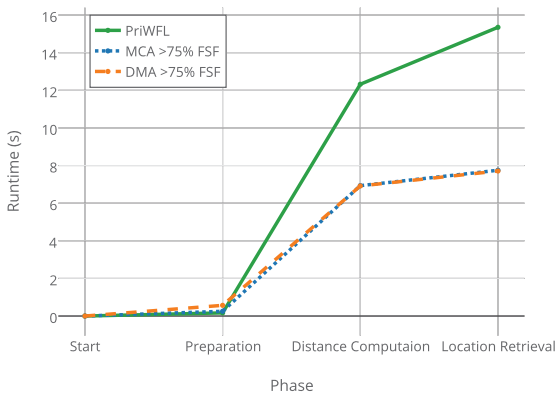

Fig. 7. PriWFL runtime comparison with MCA and DMA using FSF

VI. Conclusion

In this paper, we introduced and demonstrated a fingerprint-based localization scheme that utilizes Paillier homomorphic cryptography to ensure the privacy of the client. We deployed a crowd-sourced database and proposed two novel algorithms: MCA and DMA which can efficiently perform localization in large-scale buildings. Furthermore, we discussed three new filtering techniques that can be used to decrease the computational overhead. Our results demonstrated that MCA and DMA can significantly reduce the computational overhead compared to PriWFL.

Acknowledgment

The students Patrick Armengol and Rachelle Tobkes were supported by the US NSF REU Site at FIU. Program Grant No: REU-CNS-1461119.

References

[1] K. Chintalapudi, A. Padmanabha Iyer, and V. N. Padmanabhan, "Indoor localization without the pain," in *Proc. Int. Conf. Mobile Computing and Networking (MOBICOM)*, New York, NY, 2010, pp. 173–184. [Online]. Available: http://doi.acm.org/10.1145/1859995.1860016

[2] M. Youssef and A. Agrawala, "The Horus WLAN location determination system," in *Proc. Int. Conf. Mobile Systems, Applications, and Services (MOBSYS)*, 2005, pp. 205–218. [Online]. Available: http://doi.acm.org/10.1145/1067170.1067193

[3] P. Bahl and V. Padmanabhan, "RADAR: an in-building RF-based user location and tracking system," in *Proc. IEEE INFOCOM*, vol. 2, 2000, pp. 775–784 vol.2.

[4] M. Quan, E. Navarro, and B. Peuker, "Wi-Fi localization using RSSI fingerprinting," 2010.

[5] H. Li, L. Sun, H. Zhu, X. Lu, and X. Cheng, "Achieving privacy preservation in wifi fingerprint-based localization," in *Proc. IEEE INFOCOM*. IEEE, 2014, pp. 2337–2345.

[6] M. Gruteser and D. Grunwald, "Enhancing location privacy in wireless LAN through disposable interface identifiers: a quantitative analysis," *Mobile Networks and Applications*, vol. 10, no. 3, pp. 315–325, 2005.

[7] A. R. Beresford and F. Stajano, "Location privacy in pervasive computing," *IEEE Pervasive Computing*, no. 1, pp. 46–55, 2003.

[8] J. Freudiger, "Short: How talkative is your mobile device? An experimental study of WiFi probe requests."

[9] A. Beresford and F. Stajano, "Mix zones: user privacy in location-aware services," in *Proc. Int. Conf. Pervasive Computing and Communications Workshops*, Mar. 2004, pp. 127–131.