An earlier version of this paper will appear in the proceedings of the IEEE International Conference on Computer Communications 2018 (INFOCOM 2018).

The Death and Rebirth of Privacy-Preserving WiFi Fingerprint Localization with Paillier Encryption (Full Version)

Zheng Yang and Kimmo Järvinen

University of Helsinki, Department of Computer Science
P.O. Box 68, FI-00014, Helsinki, Finland
{zheng.yang, kimmo.u.jarvinen}@helsinki.fi

Abstract. Localization based on premeasured WiFi fingerprints is a popular method for indoor localization where satellite based positioning systems are unavailable. In these systems, privacy of the users' location is lost because the location is computed by the service provider. In INFOCOM'14, Li et al. presented PriWFL, a WiFi fingerprint localization system based on additively homomorphic Paillier encryption, that was claimed to protect both the users' location privacy and the service provider's database privacy. In this paper, we demonstrate a severe weakness in PriWFL that allows an attacker to compromise the service provider's database under a realistic attack model and also identify certain other problems in PriWFL that decrease its localization accuracy. Hence, we show that PriWFL does not solve the privacy problems of WiFi fingerprint localization. We also explore different solutions to implement secure privacy-preserving WiFi fingerprint localization and propose two schemes based on Paillier encryption which do not suffer from the weakness of PriWFL and offer the same localization accuracy as the privacy-violating schemes.

Keywords: Localization, privacy, security, WiFi fingerprint, cryptanalysis, homomorphic encryption, attack

1 Introduction

The ability to determine a user's location is essential for many contemporary applications. Global navigation satellite systems (GNSS) such as GPS are the primary technologies for obtaining the user's location. In these systems, a GNSS chip in the user's possession locally calculates its position based on signals received from satellites. Hence, GNSS fully preserves the privacy of users' locations. Unfortunately, GNSS is completely unavailable or has poor service when the user is indoors or even in certain outdoor environments (e.g., urban canyons). Premeasured databases have been proposed as solutions for accurate localization also in such cases and they have become a popular method for indoor localization.

In these solutions, a service provider first records received signal strengths (RSS) of access points (APs) in various predefined locations and stores them into a database. The APs are typically WiFi APs (see, e.g., [7, 10–12, 20, 25, 27]), but also systems based on cellular [26], RFID [5], Bluetooth [6], and Zigbee [18] signals have been proposed. A user measures the RSS values for all APs stored in the database (some of which are likely to be out of reach) in his/her location and sends this

"fingerprint" to the service provider's server hosting the database. The server uses the "fingerprint" and the database to calculate the location of the user.

Contrary to GNSS, the fingerprint-based schemes violate users' location privacy because the locations are calculated by the server. Users' locations are high-value information that may allow learning very sensitive information (e.g., regularly visited shops, bars, places of worship, etc.) and could be used for very accurate profiling, e.g., for targeted marketing. On the other hand, the service provider wants to keep its database private because it is a central business secret and also because database updates would be difficult for a distributed database. Hence, a privacy-preserving localization scheme should derive the users' locations without revealing (a) users' locations and (b) the service provider's database to the other party.

Konstantinidis *et al.* in [14] presented privacy-preserving localization based on k-anonymity, which is a well-studied problem, e.g., in privacy-preserving medical data. To simplify, their solution hides the user's real location trace among $k-1$ fake traces. The service provider is assumed not to use any auxiliary information including statistics (e.g., average numbers of users in specific areas) or even to validate the users' requests against the building map. Use of such auxiliary information allows to distinguish real traces and, consequently, to track the user's past and future movements. Hence, the solution essentially trusts the service provider to be 'honest'.

In INFOCOM 2014, Li *et al.* [16] presented a privacy-preserving WiFi fingerprint localization scheme called PriWFL and claimed that it protects both the users' locations and the database when the parties are 'honest-but-curious'; i.e., they honestly follow the protocol but can utilize any information given to them (and also auxiliary information). The scheme is based on the additively homomorphic Paillier cryptosystem [19], which allows to compute additions and subtractions with ciphertexts. The user encrypts a fingerprint with Paillier encryption, the server computes its distances to the database entries with the ciphertexts, and the user decrypts the distances and calculates its own location. Because the service provider does not have the secret keys to decrypt the users' fingerprints, PriWFL preserves the privacy of users' locations. To prevent the users from calculating the database from the distances, the server blinds their exact values with some randomness.

In this paper, we have two main contributions:

- We present an attack against PriWFL from [16]. Our attack fully discloses the service provider's database to an attacker (a user) under a realistic attack model. Our attack shows that PriWFL offers little additional protection compared to the case where the service provider gives its database to the users. We also identify certain other disadvantages of PriWFL.
- We explore certain directions to implement privacy-preserving fingerprint localization schemes. In particular, we introduce two solutions based on Paillier encryption and two different multiparty computation approaches that are secure and feasible for practical deployment.

The rest of the paper is structured as follows. Sect. 2 presents the required preliminaries. We present our attack and discuss other disadvantages of PriWFL in Sect. 3. In Sect. 4, we present new solutions for secure privacy-preserving WiFi

Table 1. WiFi Fingerprint Reference Database D

i	L_i	AP_1	AP_2	AP_3	$\cdots$	AP_N
1	(x_1, y_1, z_1)	$v_{1,1}$	$v_{1,2}$	$v_{1,3}$	$\cdots$	$v_{1,N}$
2	(x_2, y_2, z_2)	$v_{2,1}$	$v_{2,2}$	$v_{2,3}$	$\cdots$	$v_{2,N}$
3	(x_3, y_3, z_3)	$v_{3,1}$	$v_{3,2}$	$v_{3,3}$	$\cdots$	$v_{3,N}$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
M	(x_M, y_M, z_M)	$v_{M,1}$	$v_{M,2}$	$v_{M,3}$	$\cdots$	$v_{M,N}$

fingerprint localization schemes and discuss their feasibility. Finally, Sect. 5 draws conclusions.

2 Preliminaries

2.1 WiFi Fingerprint Localization

A WiFi fingerprint localization service includes two parties: a client $\mathcal{C}$, which is, e.g., a user's smartphone, and the service provider's server $\mathcal{S}$. The service utilizes signal strengths of APs distributed around the area covered by the service (e.g., a shopping mall, an exhibition center, etc.). If an AP is close to a specific location, then its signal is strong whereas if it is far away, then its signal is either weak or not available at all.

System setup and the reference database During the system setup, the service provider goes to M specific locations (x_i, y_i, z_i) for $i = 1, \ldots, M$ and measures RSS values $V_i = (v_{i,1}, v_{i,2}, \ldots, v_{i,N})$ for all N APs used in the system. A reference database is constructed using these values as follows and stored into $\mathcal{S}$:

$$D = \left\langle i, (x_i, y_i, z_i), V_i = \{v_{i,j}\}_{j=1}^{N} \right\rangle_{i=1}^{M}. \tag{1}$$

The structure of D is shown in Table 1. The service provider also publishes the table

$$T_1 = \{AP_j\}_{j=1}^{N} \tag{2}$$

where AP_j is the j-th AP's unique public identifier (e.g., its MAC address).

Location retrieval When $\mathcal{C}$ wants to know its location, it measures the RSS of all APs listed in T_1 of (2) and constructs a "fingerprint" $F = (f_1, f_2, \ldots, f_N)$ where f_j is the RSS of AP_j in $\mathcal{C}$'s location. It then sends F to $\mathcal{S}$ who finds the k-nearest neighbors of F from D by calculating the differences d_i between F and the measurements V_i in D for all $i = 1, \ldots, M$. While various distance functions can be used, we assume that d_i is the following Euclidean distance:

$$\begin{aligned} d_i = ||V_i - F||^2 &= \sum_{j=1}^{N} (v_{i,j} - f_j)^2 \\ &= \sum_{j=1}^{N} v_{i,j}^2 + \sum_{j=1}^{N} (-2 v_{i,j} f_j) + \sum_{j=1}^{N} f_j^2. \end{aligned} \tag{3}$$

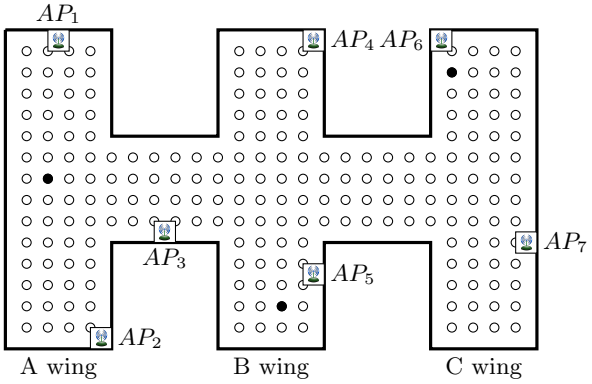

Fig. 1. An example of a WiFi fingerprint localization system for a one-story building with three wings (A, B, C) and seven APs. The white dots show the M locations in the reference database D and the black dots are highlighted locations discussed in the text.

$\mathcal{S}$ finds the indexes of the k smallest distances: $\pi_1, \pi_2, \ldots, \pi_k$ such that $d_{\pi_1} \leq d_{\pi_2} \leq \ldots \leq d_{\pi_k} \leq d_i$ for all $i \neq \pi_1, \pi_2, \ldots, \pi_k$. $\mathcal{S}$ then calculates $\mathcal{C}$'s location $L_{\mathcal{C}} = (x, y, z)$ as the centroid of the locations (x_i, y_i, z_i), for $i \in \{\pi_j\}_{j=1}^{k}$. Finally, $\mathcal{S}$ sends $L_{\mathcal{C}}$ to $\mathcal{C}$ who then knows its location.

Example Fig. 1 shows an artificial example of a one-story building with three wings (A, B, C). The building includes $N = 7$ APs and the service provider has measured their RSS values in $M = 216$ locations. For the sake of simplicity, we assume in this paper that $v_{i,j}$ are four-bit values so that $v_{i,j} = 0$ means that AP_j is unavailable at location i where as the value $v_{i,j} = 15$ is the strongest possible RSS[1]. The database entries for the highlighted locations shown in Fig. 1 could be, e.g., as follows:

$$\langle 21, (2, 7, 0), (10, 9, 12, 2, 4, 0, 0) \rangle \tag{4}$$

$$\langle 121, (13, 13, 0), (1, 0, 7, 9, 13, 0, 3) \rangle \tag{5}$$

$$\langle 162, (21, 2, 0), (0, 0, 0, 1, 5, 14, 11) \rangle \tag{6}$$

In a large building, the signal of a single AP cannot cover the whole building and, therefore, APs are unavailable in certain parts of the building; i.e., $v_{i,j} = 0$ for some i. E.g., in the above example, AP_1 and AP_2, which are located in the A wing, are not available in the C wing. Besides, two nearby locations are likely to be similar; e.g., for $\langle 22, (2, 8, 0), V_{22} \rangle$, $V_{22} \approx V_{21}$, where V_{21} is given in (4), and, hence, $v_{22,6}$ and $v_{22,7}$ are also zeros with a high probability.

2.2 Paillier Public-Key Encryption Scheme

Let $\kappa \in \mathbb{N}$ be the security parameter and $[n] = \{1, \ldots, n\} \subset \mathbb{N}$ to denote the set of integers between 1 and n. We use the notation $a \xleftarrow{\$} S$ to denote the operation which samples a uniform random element from a set S.

[1] In practice, it is more common to use power ratios in decibels (dBm) (e.g., -30 dBm is a very strong signal whereas -80 dBm implies very low WiFi functionality). A constant (e.g., $+100$ in [28]) is typically used for an AP that is 'unavailable'. Our attack works equally well also if dBm values are used.

The Paillier public-key encryption (PKE) scheme [19] is a probabilistic encryption scheme based on the decisional composite residuosity problem. Let $\mathsf{PrimG}(\kappa)$ be a function which generates a set of primes of length κ. The Paillier PKE scheme mainly consists of the following three algorithms:

- **Key Generation (KeyGen).** Given the security parameter κ, the algorithm chooses two large primes $p, q \xleftarrow{\$} \mathsf{PrimG}(\kappa/2)$, and computes $n = p \cdot q$. It also selects a group generator g for the multiplicative group $\mathbb{Z}_{n^2}^*$, such that the order of g is a non-zero multiple of n. The public key pk is a tuple (n, g) and the secret key sk is $\lambda = \mathrm{lcm}(p-1, q-1)$. This algorithm returns (pk, sk).
- **Encryption (Enc).** This algorithm takes a message $m < n$ and a public key (n, g) as inputs. It selects a random $r \xleftarrow{\$} [n-1]$, and computes the ciphertext:

$$C = g^m \cdot r^n \bmod n^2. \tag{7}$$

  The output of this algorithm is C. For simplicity, we may omit modulus n^2 in the rest of the paper.
- **Decryption (Dec).** This algorithm takes $C < n^2$ and the secret key λ as inputs and it outputs the plaintext $m = \frac{L(C^\lambda) \bmod n^2}{L(g^\lambda) \bmod n^2} \bmod n$, where $L(u) = \frac{u-1}{n}$.

Paillier PKE scheme is additively homomorphic over the group $\mathbb{Z}_n$. Namely, for two ciphertext $C_1 = \mathsf{Enc}(pk, m_1)$ and $C_1 = \mathsf{Enc}(pk, m_2)$, we have that

$$\mathsf{Dec}(sk, C_1 \cdot C_2 \bmod n^2) = m_1 + m_2 \pmod n \tag{8}$$

$$\mathsf{Dec}(sk, C_1 \cdot C_2^{-1} \bmod n^2) = m_1 - m_2 \pmod n \tag{9}$$

where the inverse can be computed via the exponentiation $C_2^{-1} = C_2^{n-1} \bmod n^2$. Using the above homomorphic additions, it is also possible to compute multiplications and divisions by a scalar t:

$$\mathsf{Dec}(sk, C_1^t \bmod n^2) = t \cdot m_1 \pmod n \tag{10}$$

$$\mathsf{Dec}(sk, C_1^{t^{-1} \bmod n} \bmod n^2) = m_1/t \pmod n \tag{11}$$

where $t^{-1} \bmod n$ can be computed with the Extended Euclidean Algorithm.

2.3 The PriWFL Scheme

In this subsection, we review the complete PriWFL scheme introduced by Li *et al.* [16]. Similarly to the basic scheme of Sect. 2.1, also the PriWFL scheme is run between $\mathcal{C}$ and $\mathcal{S}$; i.e., there are no (trusted) third parties.

System setup The system setup remains mostly the same: $\mathcal{S}$ has D as in (1) and Table 1. In addition to T_1, $\mathcal{S}$ also publishes the following table:

$$T_2 = \langle i, (x_i, y_i, z_i) \rangle_{i=1}^{M}. \tag{12}$$

When $\mathcal{C}$ subscribes to the service, it generates a key pair (sk, pk) for the Paillier cryptosystem for a sufficiently large κ (e.g., $\kappa = 2048$) and sends $pk = (n, g)$ to $\mathcal{S}$.

Location retrieval PriWFL works in three phases:

- $\mathcal{C}$ measures $F = (f_1, f_2, \ldots, f_N)$ with f_j for all AP_j listed in T_1. Instead of sending F directly to $\mathcal{S}$, $\mathcal{C}$ computes

$$C_{j,0} = \mathsf{Enc}(pk, -2f_j) \tag{13}$$

$$C_{j,1} = \mathsf{Enc}(pk, f_j^2 + u_j) \tag{14}$$

  where $u_j \xleftarrow{\$} \mathcal{R}_U$, for $j = 1, \ldots, N$; $\mathcal{R}_U$ is a randomness space of PriWFL (see Sect. 3.3 for more discussion about PriWFL randomness spaces). Then, $\mathcal{C}$ sends $\{C_{j,0}, C_{j,1}\}_{j=1}^{N}$ to $\mathcal{S}$ who cannot open the encryption because it does not have sk.
- When $\mathcal{S}$ receives $\{C_{j,0}, C_{j,1}\}_{j=1}^{N}$, it selects
  1. A random number $\tau \leq N' \leq N$, where τ is a fixed threshold (e.g., $\tau = 6$ was suggested in [16], but see Sect. 3.3 for more discussion). Using N', $\mathcal{S}$ selects a random selection set $S = \{s_1, s_2, \ldots, s'_N\}$ such that $s_i \in [N]$ and $s_i \neq s_j$ for all $i \neq j$. I.e., $\mathcal{S}$ selects a set of N' random APs from all N APs.
  2. A random offset $R \xleftarrow{\$} \mathcal{R}_R$ where $\mathcal{R}_R$ is a randomness space of PriWFL (see Sect. 3.3).

  After this, $\mathcal{S}$ computes, for $i = 1, \ldots, M$:

$$\Delta_{i,1} = \mathsf{Enc}(pk, \sum_{j \in S} v_{i,j}^2) \tag{15}$$

$$\Delta_{i,2} = \prod_{j \in S} C_{j,0}^{v_{i,j}} \tag{16}$$

$$\Delta_{i,3} = \prod_{j \in S} C_{j,1} \tag{17}$$

  The terms correspond to the encryptions of the terms required to compute the distances according to (3) so that $\Delta_{i,2} = \mathsf{Enc}(pk, \sum_{j \in S}(-2v_{i,j}f_j))$ and $\Delta_{i,3} = \mathsf{Enc}(pk, \sum_{j \in S}(f_j^2 + u_j))$. However, they have been computed by using the N' APs selected in S instead of all N APs used in (3). Next, $\mathcal{S}$ computes the encrypted distance masked by the random offset R:

$$C_{d_i+R} = \Delta_{i,1} \cdot \Delta_{i,2} \cdot \Delta_{i,3} \cdot \mathsf{Enc}(pk, R) \tag{18}$$

  After this, $\mathcal{S}$ sends $\{C_{d_i+R}\}_{i=1}^{M}$ to $\mathcal{C}$.
- When $\mathcal{C}$ receives the encrypted distances $\{C_{d_i+R}\}_{i=1}^{M}$, it uses sk to decrypt $d_i + R$ for $i = 1, \ldots, M$. Then, it finds $\pi_1, \ldots, \pi_k$, the indexes of the k smallest distances. Because each $d_i + R$ is blinded by the same offset R, their order is still preserved. It uses the public table T_2 to get (x_i, y_i, z_i), for $i \in \{\pi_j\}_{j=1}^{k}$ and, then, computes its location $L_{\mathcal{C}}$. Notice that this location calculation is similar to the basic scheme in Sect. 2.1, except that it is performed by $\mathcal{C}$ itself instead of $\mathcal{S}$ and that it is calculated with only a subset of N' APs, selected by $\mathcal{S}$.

In [16], PriWFL was claimed to protect (a) $\mathcal{C}$'s location $L_{\mathcal{C}}$ from $\mathcal{S}$ thanks to the use of Paillier encryption and randomness u_j and (b) $\mathcal{S}$'s database D from $\mathcal{C}$ thanks to the random selection set S and random offset R. In Sect. 3, we show that the second claim is not true (and that u_j is not needed to get the first claim).

2.4 Threat Model

In order to show the security problems of PriWFL, we review the same threat model that was defined in [16] where four kinds of attacks were considered under the general 'honest-but-curious' attack model:

- **Client Location Privacy Attack I (CLPA-I)**: The attacker $\mathcal{A}$ directly obtains $\mathcal{C}$'s location after intercepting $\mathcal{C}$'s queries.
- **Client Location Privacy Attack II (CLPA-II)**: $\mathcal{A}$ infers $\mathcal{C}$'s location after getting $\mathcal{C}$'s sampled WiFi fingerprints.
- **Server Data Privacy Attack I (SDPA-I)**: $\mathcal{A}$ obtains a WiFi fingerprint database D' which is identical to $\mathcal{S}$'s database D.
- **Server Data Privacy Attack II (SDPA-II)**: $\mathcal{A}$ gets a WiFi fingerprint database D' which is close to $\mathcal{S}$'s database D. Namely, D' can be used to provide a similar location service as $\mathcal{S}$'s database D.

Following the 'honest-but-curios' attack model, we assume that both $\mathcal{C}$ and $\mathcal{S}$ honestly follow the protocol specifications, but both of them may be interested in compromising the other party's private information. I.e., $\mathcal{A}$ may masquerade as either $\mathcal{C}$ or $\mathcal{S}$ in order to break the counterpart's privacy and $\mathcal{A}$ is allowed to use fabricated inputs to the protocol as long as they follow the general format and specifications of the protocol. In particular, $\mathcal{C}$ is allowed to send fabricated queries to $\mathcal{S}$ who cannot notice this because a query is encrypted with Paillier encryption in PriWFL.

3 Analysis of PriWFL

In this section, we analyze PriWFL in detail. In Sect. 3.1, we introduce an attack that implements the threat model SDPA-I, which is the stronger of the two server data privacy attacks. Consequently, our attack achieves a complete break of the server-side security of PriWFL. In Sect. 3.2, we present a new variant of the attack which allows an attacker to satisfy the preconditions of the above attack and which may work even as an independent attack if the RSS values are from a small set. Furthermore, in Sect. 3.3, we also discuss some other non-trivial issues which were overlooked in PriWFL.

3.1 A Practical SDPA-I Attack

In this subsection, we present the first main contribution of this paper: an attack that fully discloses $\mathcal{S}$'s database D under a realistic attack condition. In our attack, the attacker $\mathcal{A}$ subscribes to the system as a legitimate client $\mathcal{C}$ and faithfully follows the protocol (honest-but-curious).

Precondition for the attack We assume that $\mathcal{A}$ knows certain "special" RSS values stored in $\mathcal{S}$'s database D. Specifically, $\mathcal{A}$ must know two RSS values $v_{a,\gamma}$ and $v_{b,\gamma}$ to be able to obtain all other $v_{i,\gamma}$ for AP_γ. While this may sound as a very strong assumption, we will next show that $\mathcal{A}$ can easily obtain this information in practical settings. We assume that the building is large enough so that APs

are unavailable in parts of the building. This is a realistic assumption because typically WiFi APs cover only some tens of meters and there is no point in using a localization scheme in a very small building.

The above requirement is satisfied if $\mathcal{A}$ knows two locations where AP_γ is unavailable: $v_{a,\gamma} = v_{b,\gamma} = 0$. In PriWFL, the locations (x_i, y_i, z_i), for $i = 1, \ldots, M$, are public information given to $\mathcal{C}$ in T_2. Hence, $\mathcal{A}$ can go to any location ℓ and make RSS measurements of all APs listed in T_1. Consequently, $\mathcal{A}$ will likely find out many APs which are unavailable at this location and she has obtained the first required value for all these APs. E.g., if $\mathcal{A}$ makes the measurement in the highlighted location in the A wing of Fig. 1, then at least AP_6 and AP_7 will be unavailable. If an AP is unavailable in location ℓ, then it is unavailable with high probability also in the location ℓ' which is next to the location ℓ (the eight white dots surrounding the black dot in Fig. 1). Furthermore, this assumption is easy to verify by making a new measurement in ℓ'. Hence, the second required value is found for all APs that were unavailable in the location ℓ. On the other hand, if an AP has a very strong RSS value in the location ℓ, then $\mathcal{A}$ knows that they are close to the location ℓ and, consequently, deducts that they must be unavailable in a location ℓ'' which is far from the location ℓ. E.g., because AP_1, AP_2, and AP_3 are strong in the highlighted dot in the A wing in Fig. 1, then they must be unavailable in the C wing (e.g., the black dot and its six neighbor dots in the C wing in Fig. 1). This gives the required values for all APs with strong signals in the measurement. Hence, the required values are missing only for APs which have medium strength signals in the location ℓ. They can be obtained by making a new measurement in another part of the building (e.g., in the B wing in Fig. 1). If $\mathcal{A}$ makes an error in the above procedure, then the attack fails for the affected AP(s), but not for the entire D, and $\mathcal{A}$ can spot such errors during the attack. To summarize, $\mathcal{A}$ can obtain the required values $v_{a,\gamma} = v_{b,\gamma} = 0$ for all APs by making few measurements in a building covered by PriWFL.

The attack The attack arises because all distances calculated in a query are masked with the same randomness R (chosen by $\mathcal{S}$). We will extensively exploit the fact that the randomness R can be removed by subtracting two masked distances: $(d_i + R) - (d_j + R) = d_i - d_j$. To get the γ-th column of D, $\mathcal{A}$ makes two types of special location queries to $\mathcal{S}$ as follows:

- **All-Zero Query:** $\mathcal{A}$ generates a fake WiFi fingerprint with all 0s: $F^0 = (0, 0, 0, \ldots, 0)$. Equation (3) shows that this query yields distances which are computed with only $v_{i,j}^2$ under a random selection set S^0, i.e.,

$$d_i^0 = \sum_{z \in S^0} v_{i,z}^2 + R^0, \tag{19}$$

  where S^0 and R^0 can be different between different queries but remain the same for all i in one query.
- **Single-One Query**: $\mathcal{A}$ generates a fake WiFi fingerprint where the γ-th value is 1 and all other $N - 1$ values are 0s; e.g, $F^1 = \{0, 1, 0, ..., 0\}$ for $\gamma = 2$. This query yields distances which are computed with $v_{i,j}^2$ and $-2v_{i,j}$ under a random

selection set S^1, i.e.,

$$d_i^1 = \begin{cases} \sum_{z \in S^1} v_{i,z}^2 - 2v_{i,\gamma} + 1 + R^1, & \text{when } \gamma \in S^1 \\ \sum_{z \in S^1} v_{i,z}^2 + R^1, & \text{when } \gamma \notin S^1 \end{cases} \tag{20}$$

where S^1 and R^1 can be different between different queries but remain the same for all i in one query.

Note that $\mathcal{S}$ cannot distinguish the above special queries from the ordinary queries of an honest party because they are encrypted with Paillier encryption which is probabilistic and semantically secure. Next, we show how these queries can be used to compromise the γ-th column of $\mathcal{S}$'s database D, in particular, by finding distances that were computed using all-zero and single-one queries so that $S^0 = S^1$ and $\gamma \in S^1$. The probability for this collision of the selection sets is overwhelming after a few queries if N is not large (e.g., $N = 10$ in proof of [16, Theorem 3]).

Our attack basically has two phases: attack preparation and on-line attack, which are as follows:

Attack Preparation Phase: $\mathcal{A}$ finds two unavailable locations for each AP, according to the public tables T_1 and T_2 which are provided by $\mathcal{S}$ (see Sect. 3.1). Now $\mathcal{A}$ has an initial target database (ITD) with at least two known zeros in each column. Table 2 shows an example ITD, where each $v_{i,j} \neq 0$ is unknown to $\mathcal{A}$.

Table 2. Initial Target WiFi Fingerprints Database

i	L_i	AP_1	AP_2	AP_3	$\cdots$	AP_N
1	(x_1, y_1, z_1)	$v_{1,1}$	0	$v_{1,3}$	$\cdots$	0
2	(x_2, y_2, z_2)	$v_{2,1}$	0	0	$\cdots$	$v_{2,N}$
3	(x_3, y_3, z_3)	0	$v_{3,2}$	0	$\cdots$	0
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
M	(x_M, y_M, z_M)	0	$v_{M,2}$	0	$\cdots$	$v_{M,N}$

On-line Attack Phase: $\mathcal{A}$, who has subscribed to the system as a legitimate $\mathcal{C}$, has a public/private key pair $(pk_{\mathcal{A}}, sk_{\mathcal{A}})$ of the Paillier PKE scheme. Then, $\mathcal{A}$ does the following steps:

– **Step 1**: $\mathcal{A}$ sends several all-zero queries to $\mathcal{S}$. Specifically, $\mathcal{A}$ runs Algorithm 1 with inputs $F^0 = \{0\}_{i=1}^N$ and an integer q^0 to collect q^0 distance sets and stores them in $\mathsf{D}^0 = \{\{d_i^{0,1}\}_{i=1}^M, \{d_i^{0,2}\}_{i=1}^M, \ldots, \{d_i^{0,q^0}\}_{i=1}^M\}$.

Algorithm 1: Collect distance sets

Input: $F = \{f_j\}_{j=1}^{N}$ and q
Output: $\mathsf{D} = \{\{d_i^1\}_{i=1}^{M}, \{d_i^2\}_{i=1}^{M}, \ldots, \{d_i^q\}_{i=1}^{M}\}$

1 $\mathsf{D} \leftarrow \emptyset$;
2 **for** $\phi = 1$ **to** q **do**
3 **for** $j = 1$ **to** N **do**
4 $C_{j,0}^{\phi} \leftarrow \mathsf{Enc}(pk_{\mathcal{A}}, -2f_j)$;
5 $C_{j,1}^{\phi} \leftarrow \mathsf{Enc}(pk_{\mathcal{A}}, {f_j}^2)$;
6 **send** $\{C_{j,0}^{\phi}, C_{j,1}^{\phi}\}_{j=1}^{N}$ to $\mathcal{S}$;
7 **receive** $\{C_{d_i}^{\phi}\}_{i=1}^{M}$ from $\mathcal{S}$;
8 **for** $i = 1$ **to** M **do**
9 $d_i^{\phi} \leftarrow \mathsf{Dec}(sk_{\mathcal{A}}, C_{d_i}^{\phi})$;
10 **append** $\{d_i^{\phi}\}_{i=1}^{M}$ to D ;
11 **return** D

Because each distance set $\{d_i^t\}_{i=1}^{M} \in \mathsf{D}^0$ is related to a selection set S^t, D^0 implies a set of selection sets denoted by $\mathsf{S}^0 = \{S^{0,1}, S^{0,2}, \ldots, S^{0,q^0}\}$ (which can include duplicates). However, $\mathcal{A}$ does not need to know the exact values of N' and S^0 in our attack.

– **Step 2**: $\mathcal{A}$ sends several queries to $\mathcal{S}$ by using single-one fingerprints F^1 with $f_\gamma^1 = 1$. Specifically, $\mathcal{A}$ runs Algorithm 1 with inputs F^1 and q^1 to collect distance sets $\mathsf{D}^1 = \{\{d_i^{1,1}\}_{i=1}^{M}, \{d_i^{1,2}\}_{i=1}^{M}, \ldots, \{d_i^{1,q^1}\}_{i=1}^{M}\}$.

  Similarly, D^1 also implies a set of selection sets $\mathsf{S}^1 = \{S^{1,1}, S^{1,2}, \ldots, S^{1,q^1}\}$ which were used to compute the distance sets stored in D^1.

– **Step 3**: Next, $\mathcal{A}$ compromises the γ-th column of the database D. First, $\mathcal{A}$ finds out a distance set pair from D^0 and D^1 where both sets are computed using the same selection set. $\mathcal{A}$ uses its existing knowledge on D to check whether $\{d_i^{0,t}\}_{i=1}^{M} \in \mathsf{D}^0$ and $\{d_i^{1,z}\}_{i=1}^{M} \in \mathsf{D}^1$, for some t, z, were generated using the same selection set. Let the indexes a and b be such that $v_{a,\gamma} = v_{b,\gamma} = 0$. $\mathcal{A}$ runs Algorithm 2 with inputs D^0, D^1, a and b, to get two sorted and trimmed distance set variables $\mathsf{D}^{0\prime}$ and $\mathsf{D}^{1\prime}$. Let $|\cdot|$ be an operation that gives the cardinality of a distance set variable. Note that

$$d_a^{0,t} - d_b^{0,t} = \sum_{j \in S^{0,t}} (v_{a,j}^2 - v_{b,j}^2), \tag{21}$$

and

$$d_a^{1,z} - d_b^{1,z} = \sum_{j \in S^{1,z}} (v_{a,j}^2 - v_{b,j}^2). \tag{22}$$

Hence, having $d_a^{0,t} - d_b^{0,t} = d_a^{1,z} - d_b^{1,z}$ on Line 6 of Algorithm 2 implies that $S^{0,t} = S^{1,z}$ with overwhelming probability. After executing Algorithm 2, $\mathcal{A}$ can get two distance set variables in which, for $1 \le \nu \le |\mathsf{D}^{0\prime}|$, $\{d_i^{0,\nu}\}_{i=1}^{M} \in \mathsf{D}^{0\prime}$ and $\{d_i^{1,\nu}\}_{i=1}^{M} \in \mathsf{D}^{1\prime}$ are generated based on the same selection set, i.e., $S^{0,\nu} = S^{1,\nu}$.

Algorithm 2: Sort and trim distance sets

Input: D^0, D^1, a and b such that $v_{a,\gamma} = v_{b,\gamma} = 0$
Output: $\mathsf{D}^{0\prime}$ and $\mathsf{D}^{1\prime}$

$q^0 \leftarrow |\mathsf{D}^0|$; $q^1 \leftarrow |\mathsf{D}^1|$; $\mathsf{D}^{0\prime} \leftarrow \emptyset$; $\mathsf{D}^{1\prime} \leftarrow \emptyset$;
for $t = 1$ **to** q^0 **do**
  get $\{d_a^{0,t}, d_b^{0,t}\}$ from D^0 ;
  for $z = 1$ **to** q^1 **do**
    get $\{d_a^{1,z}, d_b^{1,z}\}$ from D^1 ;
    if $d_a^{0,t} - d_b^{0,t} = d_a^{1,z} - d_b^{1,z}$ **then**
      append $\{d_i^{0,t}\}_{i=1}^{M}$ to $\mathsf{D}^{0\prime}$;
      append $\{d_i^{1,z}\}_{i=1}^{M}$ to $\mathsf{D}^{1\prime}$;
      break ;
return $\mathsf{D}^{0\prime}, \mathsf{D}^{1\prime}$

Finally, to compromise all RSS values in the γ-th column of D, $\mathcal{A}$ runs Algorithm 3 with inputs $\mathsf{D}^{0\prime}$, $\mathsf{D}^{1\prime}$ and a. In Algorithm 3, $\mathcal{A}$ first finds a distance set (assuming to be indexed by θ) $\{d_i^{1,\theta}\}_{i=M} \in \mathsf{D}^{1\prime}$ which is computed using the γ-th column of D (i.e. $\gamma \in S^{1,\theta}$). Recall that the distances $d_i^{0,\theta} \in \mathsf{D}^{0\prime}$ and $d_i^{1,\theta} \in \mathsf{D}^{1\prime}$ with the same index θ are generated under the same selection set: $S^{0,\theta} = S^{1,\theta}$ (due to Algorithm 2). On the other hand, the all-zero and single-one queries differ only in the γ-th column. In fact, as shown in (19) and (20), if the γ-th column is not included in the computation, then the all-zero and single-one queries result in the same distances, but with different random offsets R^0 and R^1. The random offsets can be removed by computing the differences and, hence, $\mathcal{A}$ can determine whether the γ-th column was involved in the computation by evaluating the following equation:

$$d_i^{0,\theta} - d_a^{0,\theta} \stackrel{?}{=} d_i^{1,\theta} - d_a^{1,\theta}, \tag{23}$$

with $d_i^{0,\theta} \in \mathsf{D}^{0\prime}$ and $d_i^{1,\theta} \in \mathsf{D}^{1\prime}$ for $i = 1, \ldots, M$. If (23) evaluates 'false' for any i, then the γ-th column was used for calculating $\{d_i^{1,\theta}\}_{i=1}^{M}$; i.e., $\gamma \in S^{1,\theta}$. If all evaluate 'true', then it means that all $v_{i,\gamma} = 0$ and, hence, the γ-th column was not used in the calculation; i.e., $\gamma \notin S^{1,\theta}$.

Algorithm 3: Compromise the γ-th column of D

Input: $\mathsf{D}^{0'}$, $\mathsf{D}^{1'}$ and a such that $v_{a,\gamma} = 0$
Output: $\{v_{i,\gamma}\}_{i=1}^{M}$

```
q^0 ← |D^0|; found_γ ← 0; {v_{i,γ}}_{i=1}^M ← −1 ;
for θ = 1 to q^0 do
    for i = 1 to M do
        get {d_i^{0,θ}, d_a^{0,θ}} from D^{0'} ;
        get {d_i^{1,θ}, d_a^{1,θ}} from D^{1'} ;
        if d_i^{0,θ} − d_a^{0,θ} ≠ d_i^{1,θ} − d_a^{1,θ} then
            found_γ ← 1 ;
            break ;
    if found_γ = 1 then
        for i = 1 to M do
            v_{i,γ} ← (d_i^{0,θ} − d_a^{0,θ} − d_i^{1,θ} + d_a^{1,θ}) / 2 ;
        break ;
return {v_{i,γ}}_{i=1}^M ;
```

After finding $S^{0,\theta} = S^{1,\theta}$ so that $\gamma \in S^{1,\theta}$, $\mathcal{A}$ obtains all values in the γ-th column of D (including both zero and non-zero $v_{i,\gamma}$) via the following equation:

$$v_{i,\gamma} = \frac{d_i^{0,\theta} - d_a^{0,\theta} - d_i^{1,\theta} + d_a^{1,\theta}}{2}. \tag{24}$$

To obtain $\mathcal{S}$'s whole database D, $\mathcal{A}$ repeats the above procedure (Step 2 and Step 3) for all $\gamma = 1, \ldots, N$.

Analysis of the attack $\mathcal{A}$ only needs to find out a pair of distance sets $\{d_i^{0,\mu}\}_{i=1}^{M} \in \mathsf{D}^{0'}$ and $\{d_i^{1,\nu}\}_{i=1}^{M} \in \mathsf{D}^{1'}$ such that their selection sets $S^{0,\mu}$ and $S^{1,\nu}$ are equivalent and the γ-th column is involved in $S^{1,\nu}$. What is the probability of this case? This is a key problem about choosing the parameters for running our algorithms (in particular, the parameter q of Algorithm 1). Next, we show that such probability is non-negligible by showing that even a special case where N' satisfies $N' = N$ has non-negligible probability. In this case, the selection set includes all columns (all APs). Hence, we can have the following events:

- E1: there is at least one selection set $S^{0,\mu} \in \mathsf{S}^0$ which includes all N APs.
- E2: there is at least one selection set $S^{1,\nu} \in \mathsf{S}^1$ which includes all N APs.
- E3: $E2 \cap E1$;

It is not hard to see that we must have $S^{0,\mu} = S^{1,\nu}$ and $\gamma \in S^{1,\nu}$ when both events $E1$ and $E2$ occur. Suppose $\mathcal{A}$ can send at most q all-zero and single-one queries (i.e., $q^0 = q^1 = q$). Since $\mathsf{S}^{0'}$ and $\mathsf{S}^{1'}$ are selected independently, we have the following probabilities:

- $\Pr[N' = N] = \frac{1}{N-\tau}$;
- $\Pr[E1] = \Pr[E2] = 1 - (1 - \Pr[N' = N])^q$;
- $\Pr[E3] = \Pr[E1] \cdot \Pr[E2] = (1 - (\frac{N-\tau-1}{N-\tau})^q)^2$;

The probability of $E3$ basically implies a lower bound for the success probability of our attack because the real success probability is higher as also $N' < N$ such that $S^0 = S^1$ with $\gamma \in S^1$ leads to a successful attack. Fig. 2 shows the the probability of $E3$ with different number of APs N.

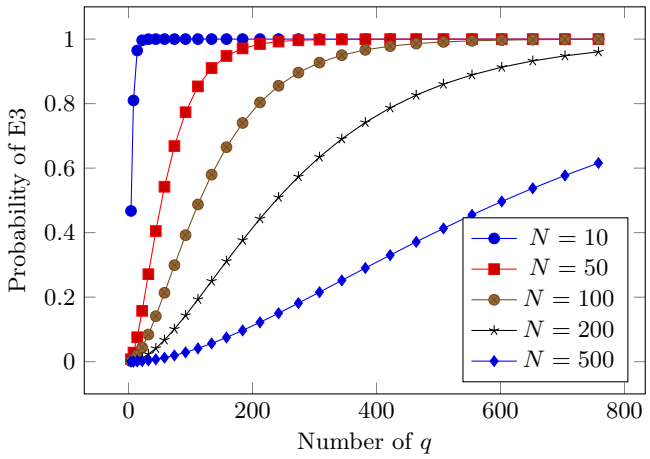

Fig. 2. Lower-bound of success probability.

With respect to $N = 10$ and $\tau = 6$ as suggested in [16], we can choose $q = 16$ to have a success probability $\Pr[E3] \approx 0.98$. Intuitively, the above attack also works for a large $N > 200$. In order obtain a high probability $\Pr[E3]$, one only needs to enlarge the number of q which is just linear in N.

It is also possible to apply the above attack idea (against PriWFL from [16]) to break the server-side security of another privacy preserving indoor location scheme which was proposed by Zhang et al. [31]. We present the concrete attack in Appendix A.

3.2 An Attack Revealing the Order of RSS Values

As discussed above, our attack relies on the existence of prior knowledge about two "zero" RSS values for each column. In the following, we show that such prior knowledge can be obtained with an attack variant that is based on the distance comparison results which lead to the leakage of the order of RSS values in a database. Even worse, we observe that such an order also leaks the range of a non-zero RSS value in $\mathcal{S}$'s database.

We now present the attack variant against the server security of PriWFL based on the idea of the above SDPA-I attack. In this attack, we mainly make use of the order of distances obtained based on All-Zero and Single-One queries. We observe that such an order leaks the order of RSS values in a column of $\mathcal{S}$'s database D that can be used to compromise all zero RSS values and the range of a non-zero RSS value in that column. Without loss of generality, we let $v_{i,j} \in \{0, 1, 2, \ldots, v_{\max}\}$ so that $v_{\max}$ is the maximum RSS value. In the following, we briefly illustrate the attack steps that allow $\mathcal{A}$ to compromise the j-th column of D:

1. $\mathcal{A}$ runs Algorithm 1 with All-Zero fingerprints $F^0 = \{0\}_{i=1}^{N}$ and an integer q^0 to get a distance set D^0.
2. $\mathcal{A}$ runs Algorithm 1 with Single-One fingerprints F^1 and an integer q^1 to get a distance set D^1.
3. $\mathcal{A}$ selects the "largest" distance set $\{d_i^{\tau,\rho^\tau}\}_{i=1}^{M}$ from D^τ (for $\tau \in \{0,1\}$) such that, when given two fixed indexes a and b, the distance difference $d_a^{\tau,\rho^\tau} - d_b^{\tau,\rho^\tau}$ is the largest among all differences $\{d_a^{\tau,i} - d_b^{\tau,i}\}_{i=1}^{q^\tau}$. The largest distance difference $d_a^{\tau,\rho^\tau} - d_b^{\tau,\rho^\tau}$ implies that the set $\{d_i^{\tau,\rho^\tau}\}_{i=1}^{M}$ is computed based on the largest selection set, i.e., $N = N'$, with overwhelming probability due to the results shown in Fig. 2.
4. $\mathcal{A}$ now has each distance $d_i^{0,\rho^0} = \sum_{t=1}^{N} v_{i,t}^2 + R^0$ and $d_i^{1,\rho^1} = -2v_{i,j} + 1 + \sum_{t=1}^{N} v_{i,t}^2 + R^1$. For arbitrary indexes a and b, we let $d_{a-b}^{0,\rho^0} = d_a^{0,\rho^0} - d_b^{0,\rho^0} = \sum_{t=1}^{N} v_{a,t}^2 - v_{b,t}^2$ and $d_{a-b}^{1,\rho^1} = d_a^{1,\rho^1} - d_b^{1,\rho^1} = -2v_{a,j} + 2v_{b,j} + \sum_{t=1}^{N} v_{a,t}^2 - v_{b,t}^2$. $\mathcal{A}$ can compute $2v_{b,j} - 2v_{a,j} = d_{a-b}^{1,\rho^1} - d_{a-b}^{0,\rho^0}$ to know which RSS value is larger. By repeating the above steps with difference indexes, $\mathcal{A}$ can obtain the order of the RSS values in the j-th column, for example:

$$0 \leq v_{128,j} = v_{111,j} = v_{99,j} = \ldots = v_{1,j} < v_{500,j} = v_{232,j} < \ldots < v_{400,j} \leq v_{\max}.$$

5. From the above order of RSS values in the j-th column, we have the following results:
   - The indexes of zero RSS values from the set of "smallest" ones, due to the large ratio of zero RSS values in a typical database (see, e.g., [17]);
   - The range of a RSS value $v_{i,j}$. Let nl (nr) be the number of distinct RSS values which are smaller (larger) than $v_{i,j}$ in the above ordered RSS values. Then, we have that $v_{i,j}$ is in the range $nl \leq v_{i,j} \leq v_{\max} - nr$.

When the RSS space is small (e.g., 4 bits) and M is large, an attacker $\mathcal{A}$ may very likely be able to fake an RSS value so that it is very close to (or even equivalent with) the real one in the target database. Hence, this attack may help $\mathcal{A}$ to lunch an SDPA-II attack.

3.3 Other Non-trivial Problems

In this subsection, we point out other non-trivial problems of PriWFL. While these problems do not have a direct effect on the security of PriWFL (as the major weakness that we revealed in Sect. 3.1), these problems may dramatically affect the localization accuracy, even up to a point that makes PriWFL unusable in practice. The problems are mainly caused by the randomly chosen values and selection set. Recall that there are different types of random selections in PriWFL (see Sect. 2.3). $\mathcal{C}$ chooses N random values $\{u_1, u_2, \ldots, u_N\}$ to blind the squared values of its fingerprint: ${f_j}^2 + u_j$. $\mathcal{S}$ chooses a random offset R and a selection set S with N' random APs. In the following, we discuss the problems related to these random selections in detail.

Problems originating from the random selection set The random selection sets cause random localization errors because distances will be calculated by using only the RSS values of the APs in the selection sets. It is very likely that some *significant* values (i.e., strong RSS values) of either $\mathcal{C}$'s F or $\mathcal{S}$'s D will be excluded from the calculation. Therefore, the accuracy of the localization service will decrease due to the random selection set. In [16], they argued that localization accuracy remains good if N' satisfies $\tau \leq N' \leq N$ with a threshold $\tau = 6$; the deduction in [16] assumed that N is small (e.g., $N = 10$). By observing certain publicly available research-oriented WiFi fingerprint databases (e.g., [28, 17]), we notice that, in practice, $N \gg 10$ (e.g., $N > 200$) and many values in D are $v_{i,j} = 0$ ("AP unavailable")[2]. In such cases, the argument of [16] is no longer valid and severe increase of localization errors can be expected to happen as a result of random selection sets.

Problems originating from the other randomness The randomness spaces $\mathcal{R}_U$ and $\mathcal{R}_R$, from which $\{u_1, u_2, \ldots, u_N\}$ and R are drawn, respectively, are not defined in [16]. If an implementer chooses the randomness space $\mathcal{R}_R$ inappropriately, it may result in random localization errors. The message space of the Paillier PKE scheme is $\mathbb{Z}_n$ (integers between 0 and $n-1$) and if a result of an operation with ciphertexts exceeds this range, then it gets reduced modulo n when decrypted. E.g., if we have $m_1 = 2$ and $m_2 = n - 1$ and we compute $\mathsf{Dec}(sk, \mathsf{Enc}(pk, m_1) \cdot \mathsf{Enc}(pk, m_2))$, then we get 1 as an output instead of $n + 1$. Hence, if $\mathcal{R}_R$ is defined so that R can be close to n, then it may happen that, for some distances d_i and d_j such that $d_i < d_j$, an "overflow" occurs for d_j and $d_i + R > d_j + R \pmod{n}$. This will have a severe effect on calculating the location because $\mathcal{C}$ does not know R and, consequently, incorrect locations (x_i, y_i, z_i) will be chosen as the k smallest distances.

The random values $\{u_1, u_2, \ldots, u_N\}$ drawn from the randomness space $\mathcal{R}_U$ do not seem to serve any real purpose because the Paillier PKE scheme is already probabilistic: if one encrypts m_1 twice, then the ciphertexts will be different even without u_i because a random r is used for every encryption as shown in (7). Hence, $\{u_j\}_{j=1}^{N}$ are not needed to protect $\mathcal{C}$'s location. They also cannot protect $\mathcal{S}$'s database because $\mathcal{C}$ can freely choose u_j (e.g., $u_j = 0$ for all j).

Summary PriWFL is both insecure and unsuitable for practical use. Our attack breaks PriWFL for all practical values of N but is particularly efficient for small N that were considered in [16]. Even if PriWFL could be fixed against the attack of Sect. 3.1, the problems with localization accuracy caused by the random selection set would still prevent its use when N is large. Hence, we believe that PriWFL is fundamentally flawed and new directions need to be taken in order to implement

[2] E.g., [17] includes a WiFi fingerprint database (BUILDING1_NEW) which is measured from a four-story building so that $M = 505$ and $N = 241$. In that database, 85.4 % of all values of D are "AP unavailable" ($v_{i,j} = 0$). For specific locations in D, the number of available APs varies from 11 to 67. Hence, most APs are unavailable in any specific location. This validates both the feasibility of the precondition of our attack (see Sect. 3.1) and the above claim about the unsuitability of PriWFL for practical use cases.

a secure privacy-preserving WiFi fingerprint localization scheme. In Sect. 4, we explore certain possible directions to achieve this ambitious goal.

4 Solutions

In this section, we explore four solutions to implement a secure privacy-preserving WiFi fingerprint localization scheme and discuss their feasibility for practical use.

4.1 Fully Homomorphic Encryption

Conceptually the most straightforward solution would be to use Fully Homomorphic Encryption (FHE), first introduced in Gentry's seminal work [9] in 2009. FHE allows arbitrary computations (both additions and multiplications) with ciphertexts and, consequently, allows $\mathcal{S}$ to calculate $\mathcal{C}$'s location $L_{\mathcal{C}}$ homomorphically in the encrypted domain without learning anything about $\mathcal{C}$'s fingerprint. Unfortunately, the excessive cost of FHE prevents its use in (almost) all practical use cases.

Even Somewhat (Levelled) Homomorphic Encryption (SHE) schemes that allow evaluating arbitrary functions up to certain predefined complexity (number of multiplications) are too complex for our use case. Lepoint and Naehrig [15] compared two SHE schemes, YASHE (now broken [1]) and FV [8], and demonstrated that using FV to homomorphically compute one execution of a lightweight SIMON-32/64 block cipher requires 3062 s (51 min) on a 4-core Intel Core i7-2600 processor at 3.4 GHz. Computations required by WiFi fingerprint localization are significantly more complex than SIMON-32/64 and, hence, we conclude that even SHE is impractical.

4.2 Secure Multiparty Computation with Garbled Circuits

In a secure multiparty computation (MPC) protocol, two parties jointly evaluate a function $f(x, y)$ without revealing their respective inputs x and y to each others. In an MPC protocol using Yao's garbled circuits (GC) [29], a party called the generator $\mathcal{G}$ generates a boolean GC $\tilde{f}$ for $f(x, y)$ and send it together with its own garbled input $\tilde{x}$ to the other party called the evaluator $\mathcal{E}$. Now, $\mathcal{E}$ obtains its garbled input $\tilde{y}$ from $\mathcal{G}$ via an oblivious transfer (OT) extension protocol, which ensures that $\mathcal{G}$ does not learn y, and then evaluates $\tilde{f}(\tilde{x}, \tilde{y})$ and receives the result. An OT extension protocol can be computed with cheap secret-key cryptography by precomputing PKE operations [2] and, thus, it adds only a small overhead about symmetric-key computations at the online phase.

It is easy to see that this MPC protocol can be used for privacy-preserving WiFi fingerprint localization if $\mathcal{S}$ is $\mathcal{G}$ with $x = D$ and $\mathcal{C}$ is $\mathcal{E}$ with $y = F$. Evaluating a GC requires only secret-key cryptography, which is computationally cheap (compared to PKE). The only problem is the excessive communication overhead that is caused, in particular, by the size of D. In the above protocol, each bit of x (and y) is replaced by κ random bits. If we assume $\kappa = 128$ (corresponds to, e.g., AES-128), the database from [17] (with $N = 241$ and $M = 505$) and that $v_{i,j}$ are encoded as four-bit values, then the size of only $\tilde{x}$ will be about 7.4 MB. Communicating $\tilde{y}$

and, especially, $\tilde{f}$ will still significantly add to this overhead (2κ bits for each non-XOR gate in $\tilde{f}$ [13, 30]). Hence, using straightforward GC-based MPC for privacy-preserving localization suffers from high communication cost which decreases its practical feasibility.

4.3 Paillier PKE scheme and the Signs of Differences

The following presents a solution relying on Paillier PKE scheme. The idea is to let $\mathcal{C}$ learn the signs of $\delta_{i,j} = d_i - d_j$ but nothing else about their values. This allows $\mathcal{C}$ to obtain the sorting of the distances and, consequently, to find the indexes $\pi_1, \ldots, \pi_k$ of the k smallest distances (those with most minus signs) without revealing other information about distances.

The protocol was inspired by [23] and works as follows. First, $\mathcal{S}$ computes the differences of all distance pairs by computing $C_{\delta_{i,j}} = C_{d_i}/C_{d_j}$ for all $1 \leq i, j \leq M$ such that $i < j$ and, then, $\mathcal{S}$ aligns the differences (via homomorphic scalar multiplications by 2^t) so that a sign of a difference is given by the t-th bit of the aligned difference. After this, the protocol repeats the following steps. $\mathcal{S}$ masks a difference with a (large) random mask and sends the result to $\mathcal{C}$ who decrypts the ciphertext and receives the masked difference. $\mathcal{C}$ then takes the LSB of a masked difference, encrypts it, and sends it back to $\mathcal{S}$. When $\mathcal{S}$ receives the encryption of the masked LSB, it homomorphically removes the LSB of the mask from it by computing a homomorphic XOR (via $a \oplus b = a + b - 2ab$) and receives the encryption of the LSB of the difference. Now, $\mathcal{S}$ can subtract this LSB from the full difference and, then, divide the value homomorphically by two (because the LSB is now guaranteed to be zero). $\mathcal{C}$ and $\mathcal{S}$ repeat the above procedure $t-1$ times to remove the $t-1$ LSBs from the aligned difference leaving only the t-th bit (the sign). Finally, $\mathcal{S}$ sends the sign bit to $\mathcal{C}$ without a mask and $\mathcal{C}$ knows which of d_i and d_j is larger.

With respect to the security of this solution under the semi-honest setting, $\mathcal{C}$'s location privacy is protected by Paillier encryption. Whereas, $\mathcal{S}$'s privacy is guaranteed by the freshly chosen large random values, and the security of the LSB sub-protocol for privately calculating the sign bits. We refer the reader to [23] for more details on the security analysis of the LSB sub-protocol. It is straightforward to see that the sign bits alone do not directly help $\mathcal{C}$ to compromise D. But for security consideration, one may need to ensure that each reference RSS value in D has a large bit-length.

The communication overhead of this solution grows quickly with M because the number of differences $\delta_{i,j}$ is $M(M-1)/2$. Also in this case, multiple differences can be packed into a ciphertext. Another important factor is the precision of $\delta_{i,j}$ because a high precision equals large t and requires multiple protocol rounds to reach the sign bit. Hence, this solution can be feasible only in specific cases (with small N and M).

Nevertheless, the above solution may be susceptible to the order attack shown in Section 4. In order to mitigate the order attack, we suggest to use a large RSS space to protect the non-zero RSS values and reduce the identical RSS values in a column.

4.4 Paillier PKE scheme and Garbled Circuits

A combination of Paillier encryption and garbled circuits can be used to solve the problem of privacy-preserving WiFi fingerprint localization by adapting, e.g., Sadeghi et al.'s solution for privacy-preserving face recognition [21] and Blanton and Gasti's solution for privacy-preserving iris and fingerprint identification [4]. In this hybrid solution, $\mathcal{C}$ encrypts the RSS values using Paillier encryption with (13) from Sect. 2.3. $\mathcal{S}$ calculates the distances by computing $C_{d_i} = \Delta_{i,1} \cdot \Delta_{i,2} \cdot \Delta_{i,3}$ by using (15)–(17) with $S = [N]$, i.e., with all APs. Because all APs are always used, $\Delta_{i,3}$ depends only on $\mathcal{C}$'s inputs and is the same for all i and, hence, it can be computed by $\mathcal{C}$: $\Delta_3 = \mathsf{Enc}(pk, \sum_{j=1}^{N} f_j)$. Now, $\mathcal{S}$ packs t distances into one ciphertext by computing $C_{\text{comb}} = \prod_{i=1}^{t} C_{d_i}^{2^{(i-1)m}}$, where m is the maximum bit-length of d_i. To prevent $\mathcal{C}$ from obtaining these distances, $\mathcal{S}$ selects a random mask $R \xleftarrow{\$} \mathcal{R}_R = [n-1]$ and computes $C_{\text{m-comb}} = C_{\text{comb}} \cdot \mathsf{Enc}(pk, R)$. Let T denote the number of ciphertexts needed to pack all M distances. E.g., if n is a 2048-bit value and $m = 16$, then we can fit $t = 127$ distances in one ciphertext. Consequently, if $N = 241$ and $M = 505$ as in [17], the above Paillier encryption part requires communication of only $N + 1 = 242$ ciphertexts (121 kB) from $\mathcal{C}$ to $\mathcal{S}$ and $T = 4$ ciphertexts (2 kB) from $\mathcal{S}$ to $\mathcal{C}$.

Upon receiving all T ciphertexts, $\mathcal{C}$ opens the Paillier encryption with sk and retrieves the masked combined distances. To remove the mask R and to securely find the k smallest distances (i.e., so that $\mathcal{C}$ does not learn d_i), $\mathcal{C}$ and $\mathcal{S}$ run a GC-based MPC protocol, where $x = R$, $y = \mathsf{Dec}(sk, C_{\text{m-comb}})$, and $f(x, y)$ is such that it first computes $y - x \pmod{n}$ (i.e., removes the mask) and, then, finds the k smallest distances. When $\mathcal{C}$ has evaluated $\tilde{f}(\tilde{x}, \tilde{y})$, it has the indexes $\pi_1, \ldots, \pi_k$ of the k smallest d_i and it can calculate its location similarly as in PriWFL. Songhori *et al.* [24] presented a memory-efficient sequential garbled gate for k-nearest neighbors search and their circuit can be used for our purpose. The communication overhead of transferring $\tilde{x}$ and $\tilde{y}$ grows linearly with M and is in the magnitude of some kBs for $M = 505$ and $\kappa = 128$. The communication overhead of the GC depends on k, M, m, n, and κ, but can be estimated from [22] and [24, Table 1] to be about 1 MB for the above parameters.

Theorem 1. *Suppose that Paillier encryption and MPC schemes are both secure. Then, the above solution resists CLPA-I and CLPA-II.*

Proof. (Sketch) Resilience against CLPA-II implies resilience against CLPA-I. Generally speaking, $\mathcal{C}$'s location privacy is guaranteed by the security properties of Paillier encryption and MPC schemes. Without the secret key of $\mathcal{C}$, $\mathcal{S}$ (or any passive adversary) is unable to infer $\mathcal{C}$'s location based on the encrypted location query and the corresponding encrypted response $C_{\text{m-comb}}$. Furthermore, $\mathcal{S}$ who produces the GC does not have access to $\mathcal{C}$'s actual inputs, due to the OT protocol, or to the output the circuit. These facts protect $\mathcal{C}$'s location privacy from $\mathcal{S}$. The formal security definitions and analysis of GC-based MPC can be found in [3].

Theorem 2. *Suppose that the MPC scheme is secure and $\mathcal{R}_R$ is large. Then, the above solution resists SDPA-I and SDPA-II.*

Proof. (Sketch) It is sufficient to show that our solution leaks no information about D to $\mathcal{A}$. The freshly chosen randomness $R \xleftarrow{\$} [n-1]$ prevents $\mathcal{A}$ from learning the combined distance. Since a modular n operation is implicitly involved in the blinded distance (so that possible "overflows" are handled in the GC), $y = \mathsf{Dec}(sk, C_{\text{m-comb}})$ is statistically close to a random value. In a nutshell, $\mathcal{C}$ cannot gain non-negligible advantage to compromise the combined distances and $\mathcal{S}$'s database. In addition, the non-zero $v_{i,j}$ in D easily sum up to thousands of unknown bits in practice. E.g., [17] contains over 70,000 bits for non-zero $v_{i,j}$ if they are four-bit values ($N = 241$, $M = 505$, and 85.4 % of values are zeros). Therefore, it would be very hard for $\mathcal{A}$ to compromise even half of these non-zero values (to get a similar database).

5 Conclusion

In this paper, we showed that PriWFL, a privacy-preserving WiFi fingerprint localization scheme presented in [16], has a severe weakness that allows an attacker, who is using the service as a legitimate client, to obtain the exact database of the service. Hence, PriWFL does not offer any protection for the service provider which renders the scheme useless in practice. We also identified certain other problems which make PriWFL unpractical especially for large N.

Because of the complete break of PriWFL, there is a need for new secure privacy-preserving WiFi fingerprint localization schemes. We explored certain solutions to implement such a scheme. All of them introduce significant communication and computation overheads compared to the basic privacy-violating scheme (see Sect. 2.1) and often also to PriWFL. In particular, we sketched two solutions based on combining Paillier encryption with a scheme, which allows the client to learn only the signs of distance differences, or either with garbled circuits. Especially, the latter solution is a promising candidate for achieving both secure and practical privacy-preserving WiFi fingerprint localization and we plan to study it (and possible other solutions) in the future. This future work includes both optimizing the preliminary schemes as well as testing them in practice by integrating them into real indoor localization systems using WiFi fingerprints.

References

1. Martin Albrecht, Shi Bai, and Léo Ducas. A subfield lattice attack on overstretched NTRU assumptions: Cryptanalysis of some FHE and graded encoding schemes. Cryptology ePrint Archive, Report 2016/127, 2016.
2. Gilad Asharov, Yehuda Lindell, Thomas Schneider, and Michael Zohner. More efficient oblivious transfer and extensions for faster secure computation. In *Proc. CCS 2013*, pages 535–548. ACM, 2013.
3. Mihir Bellare, Viet Tung Hoang, and Phillip Rogaway. Foundations of garbled circuits. In *Proc. CCS 2012*, pages 784–796. ACM, 2012.
4. Marina Blanton and Paolo Gasti. Secure and efficient protocols for iris and fingerprint identification. In *Proc. ESORICS 2011*, volume 6879 of *LNCS*, pages 190–209. Springer, 2011.
5. Kirti Chawla, Christopher McFarland, Gabriel Robins, and Connor Shope. Real-time RFID localization using RSS. In *Proc. ICL-GNSS 2013*, pages 1–6. IEEE, 2013.
6. Liang Chen, Heidi Kuusniemi, Yuwei Chen, Ling Pei, Tuomo Kröger, and Ruizhi Chen. Information filter with speed detection for indoor Bluetooth positioning. In *Proc. ICL-GNSS 2011*, pages 47–52. IEEE, 2011.

7. Eiman Elnahrawy, Xiaoyan Li, and Richard P Martin. The limits of localization using signal strength: A comparative study. In *Proc. SECON 2004*, pages 406–414. IEEE, 2004.
8. Junfeng Fan and Frederik Vercauteren. Somewhat practical fully homomorphic encryption. Cryptology ePrint Archive, Report 2012/144, 2012.
9. Craig Gentry. Fully homomorphic encryption using ideal lattices. In *Proc. STOC 2009*, pages 169–178, 2009.
10. Ville Honkavirta, Tommi Perala, Simo Ali-Loytty, and Robert Piché. A comparative survey of wlan location fingerprinting methods. In *Proc. WPNC 2009*, pages 243–251. IEEE, 2009.
11. AKM Mahtab Hossain and Wee-Seng Soh. Cramer-Rao bound analysis of localization using signal strength difference as location fingerprint. In *Proc. INFOCOM 2010*, pages 1–9. IEEE, 2010.
12. Kamol Kaemarungsi and Prashant Krishnamurthy. Modeling of indoor positioning systems based on location fingerprinting. In *Proc. INFOCOM 2004*, volume 2, pages 1012–1022. IEEE, 2004.
13. Vladimir Kolesnikov and Thomas Schneider. Improved garbled circuit: Free XOR gates and applications. In *Proc. ICALP 2008*, volume 5126 of *LNCS*, pages 486–498. Springer, 2008.
14. Andreas Konstantinidis, Georgios Chatzimilioudis, Demetrios Zeinalipour-Yazti, Paschalis Mpeis, Nikos Pelekis, and Yannis Theodoridis. Privacy-preserving indoor localization on smartphones. *IEEE Trans. Knowl. Data Eng.*, 27(11):3042–3055, November 2015.
15. Tancrède Lepoint and Michael Naehrig. A comparison of the homomorphic encryption schemes FV and YASHE. In *AFRICACRYPT 2014*, volume 8469 of *LNCS*, pages 318–335. Springer, 2014.
16. Hong Li, Limin Sun, Haojin Zhu, Xiang Lu, and Xiuzhen Cheng. Achieving privacy preservation in wifi fingerprint-based localization. In *Proc. INFOCOM 2014*, pages 2337–2345, April 2014.
17. Elena Simona Lohan *et al.* Indoor WLAN measurement data. Online: `http://www.cs.tut.fi/tlt/pos/MEASUREMENTS_WLAN_FOR_WEB.zip` (accessed: Jul. 2017), 2014.
18. Angela Song-Ie Noh, Woong Jae Lee, and Jin Young Ye. Comparison of the mechanisms of the Zigbee's indoor localization algorithm. In *Proc. SNPD 2008*, pages 13–18. IEEE, 2008.
19. Pascal Paillier. Public-key cryptosystems based on composite degree residuosity classes. In *EUROCRYPT 1999*, volume 1592 of *LNCS*, pages 223–238. Springer, 1999.
20. Teemu Roos, Petri Myllymäki, Henry Tirri, Pauli Misikangas, and Juha Sievänen. A probabilistic approach to wlan user location estimation. *Int. J. Wireless Inform. Network*, 9(3):155–164, 2002.
21. Ahmad-Reza Sadeghi, Thomas Schneider, and Immo Wehrenberg. Efficient privacy-preserving face recognition. In *Proc. ICISC 2009*, volume 5984 of *LNCS*, pages 229–244. Springer, 2009.
22. Thomas Schneider. *Engineering Secure Two-Party Computation Protocols*. PhD thesis, Ruhr-University Bochum, 2011.
23. Berry Schoenmakers and Pim Tuyls. Efficient binary conversion for Paillier encrypted values. In *EUROCRYPT 2006*, volume 4004 of *LNCS*, pages 522–537. Springer, 2006.
24. Ebrahim M. Songhori, Siam U. Hussain, Ahmad-Reza Sadeghi, and Farinaz Koushanfar. Compacting privacy-preserving k-nearest neighbor search using logic synthesis. In *Proc. DAC 2015*, pages 36:1–36:6. ACM, 2015.
25. Nattapong Swangmuang and Prashant Krishnamurthy. Location fingerprint analyses toward efficient indoor positioning. In *Proc. PerCom 2008*, pages 100–109. IEEE, 2008.
26. Jukka Talvitie and Elena Simona Lohan. Modeling received signal strength measurements for cellular network based positioning. In *Proc. ICL-GNSS 2013*, pages 1–6. IEEE, 2013.
27. Jukka Talvitie, Markku Renfors, and Elena Simona Lohan. Distance-based interpolation and extrapolation methods for RSS-based localization with indoor wireless signals. *IEEE Trans. Veh. Technol.*, 64(4):1340–1353, 2015.
28. Joaquín Torres-Sospedra *et al.* UJIIndoorLoc data set. Online: `https://archive.ics.uci.edu/ml/datasets/ujiindoorloc` (accessed: Jul. 2017), 2014.
29. Andrew C-C Yao. How to generate and exchange secrets. In *Proc. FOCS 1986*, pages 162–167. IEEE, 1986.
30. Samee Zahur, Mike Rosulek, and David Evans. Two halves makes a whole — reducing data transfer in garbled circuits using half gates. In *EUROCRYPT 2015*, volume 9057 of *LNCS*, pages 220–250. Springer, 2015.
31. Tao Zhang, Sherman S. M. Chow, Zhe Zhou, and Ming Li. *Privacy-Preserving Wi-Fi Fingerprinting Indoor Localization*, volume 9836 of *LNCS*, pages 215–233. Springer International Publishing, 2016.

A A Chosen Fingerprint Attack against Zhang et al. Scheme

In this appendix, we show how to apply the attack idea presented in Sect. 3.1 to break Zhang et al.'s scheme [31] which relies on machine learning approaches for location calculation. We first briefly review the basic construction of Zhang et al. scheme (based on the dot product scenario as an example).

To provide indoor localization, a server $\mathcal{S}$ first collects a set of WiFi fingerprints used as a training data set. The server next trains a decision algorithm with all APs in the area and obtains a database $D = (\mathbf{w}_{\mathsf{x}}, b_x, \mathbf{w}_{\mathsf{y}}, b_y)$, where $\mathbf{w}_{\mathsf{x}} = \{w_{x,i}\}_{i=1}^{N}$. A client $\mathcal{C}$ measures a fingerprint F and encrypts F using its own public key pk through Paillier PKE scheme. The encrypted fingerprint is sent to $\mathcal{S}$. Upon receiving the encryptions, $\mathcal{S}$ chooses a subset of indexes $I \subseteq [N]$ with $|I| \geq \tau$ (a predefined threshold) and uses $\mathbf{w}_{\mathsf{x}I}, \mathbf{w}_{\mathsf{y}I}$ to compute the location relying on the additively homomorphic operations of Paillier PKE. The location is basically computed via the following equation:

$$x = \langle \mathbf{w}_{\mathsf{x}}, F \rangle_I + b_x; \ \textit{and}\ y = \langle \mathbf{w}_{\mathsf{y}}, F \rangle_I + b_y. \tag{25}$$

For simplicity, we here ignore the computation steps relevant to the encryptions that do not affect the attack.

For the attack, we first review two types of special fingerprints:

- **All-Zero Query:** a Wifi fingerprint with all 0s: $F^0 = (0, 0, 0, \ldots, 0)$.
- **Single-One Query**: a Wifi fingerprint where the γ-th value is 1 and all other $N - 1$ values are 0s; e.g, $F^1 = \{0, 1, 0, ..., 0\}$ for $\gamma = 2$.

Intuitively, an All-Zero Query lets an attacker to learn b_x, b_y for free and Single-One queries reveal each $w_{x,i}$. The concrete attack steps are illustrated as follows:

- Ask an All-Zero Query F^0 to obtain a location (x_0, y_0), and obtain $b_x = x_0$ and $b_y = y_0$. Because $\langle \mathbf{w}_{\mathsf{x}}, F \rangle_I = \langle \mathbf{w}_{\mathsf{y}}, F \rangle_I = 0$ for any I.
- Ask a polynomial number of Single-One queries $F_i^1 = (0, 0, 0, f_i = 1, \ldots, 0)$ to get each $w_{x,i}$. Note that the probability of $i \in I_j$ (where I_j is the subset chosen within the j-th Single-One Query with F_i^1) is larger than τ/N (which is non-negligible). If $i \notin I_j$, we have $x_j = b_x$, where x_j is the location obtained in the j-th Single-One query. To learn whether $i \in I_j$, it is sufficient to test whether $x_j = b_x$ or not. When $i \in I_j$, we can compute $w_{x_i} = x_j - b_x$ and $w_{y_j} = y_j - b_y$. By repeating the above procedure, we can obtain the whole $\mathbf{w}_{\mathsf{x}}$ and $\mathbf{w}_{\mathsf{y}}$.

A reader can try to apply the above attack idea to break the Zhang et al. scheme also with other kernel functions.