Towards Modeling Privacy in WiFi Fingerprinting Indoor Localization and its Application

Zheng Yang[1]* and Kimmo Järvinen[2]
[1]*Singapore University of Technology and Design, 487372 Singapore*
zheng_yang@sutd.edu.sg
[2]*University of Helsinki, 00014 Finland*
kimmo.u.jarvinen@helsinki.fi

Received: November 26, 2018; Accepted: March 5, 2019; Published: March 31, 2019

Abstract

In this paper, we study privacy models for privacy-preserving WiFi fingerprint based indoor localization (PPIL) schemes. We show that many existing models are insufficient and make unrealistic assumptions regarding adversaries' power. To cover the state-of-the-art practical attacks, we propose the first formal security model which formulates the security goals of both client-side and server-side privacy beyond the curious-but-honest setting. In particular, our model considers various malicious behaviors such as exposing secrets of principles, choosing malicious WiFi fingerprints in location queries, and specifying the location area of a target client. Furthermore, we formulate the client-side privacy in an indistinguishability manner where an adversary is required to distinguish a client's real location from a random one. The server-side privacy requires that adversaries cannot generate a fabricate database which provides a similar function to the real database of the server. In particular, we formally define the *similarity* between databases with a ball approach that has not been formalized before. We show the validity and applicability of our model by applying it to analyze the security of an existing PPIL protocol. We also design experiments to test the server-privacy in the presence of database leakage, based on a candidate server-privacy attack.

Keywords: Indoor localization, WiFi fingerprint, Security Model, Privacy

1 Introduction

People spend significant amounts of their time in public indoor environments including shopping malls, libraries, airports, university campuses, etc. This has boosted the interest towards various indoor location-based applications[1, 2] such as indoor-navigation or elderly assistance and emergency responding. However, in an indoor environment, the traditional Global Positioning System (GPS) may be not available due to weak signal strengths caused by blocking constructions. To obtain a location in a building, a client has to rely on certain *indoor location services* (ILS) provided by some server of the building. The most widely used approach for ILS is the one based on the WiFi fingerprinting technique [3, 4, 5, 6, 7, 8, 9, 10, 11]. This method is very effective and popular because it uses an existing WiFi infrastructure of a building. For a WiFi fingerprint based ILS, the server holds a geo-location database (e.g. [12, Table 1]) containing signal strengths of WiFi access points (AP) in various reference locations, as explained in Section 3. Roughly speaking, a client measures the signal strengths of WiFi APs in the

Journal of Wireless Mobile Networks, Ubiquitous Computing, and Dependable Applications, 10:1 (March 2019), pp. 4-22
DOI: 10.22667/JOWUA.2019.03.31.004

*Corresponding author: ITrust, Singapore University of Technology and Design, 487372, Singapore, email: zheng.yang@rub.de, Tel: +65 84517856, Web: https://www.researchgate.net/profile/Zheng_Yang17

client's current (unknown) location and send them to the server. The server calculates the client's location based on the geo-location database, e.g., by calculating the k-nearest Euclidean distances between the client's input and reference fingerprints in the database. Finally, the server sends the location to the client. However, this naive solution cannot prevent a malicious server from tracking its clients' locations, which of course violates the clients' privacy.

Recently, several solutions, e.g. [13, 14, 15, 12], have been proposed to protect the clients' location privacy in ILSs. However, only a few pieces of research (e.g. [15]) have included a formal security model for privacy-preserving indoor localization (PPIL) schemes. This deficiency has resulted in the development of flawed protocols (e.g. [13, 15]) which may take years to discover. Therefore, applying PPIL schemes without rigorous security proofs is inherently risky. For example, in INFOCOM 2014, Li et al. [13] presented a WiFi fingerprint localization system called PriWFL which was claimed to provide both clients' location privacy and server's database privacy (which will be referred to as client-privacy and server-privacy for short, respectively). PriWFL is based on the 'honest-but-curious' setting where the adversary does not change the protocol execution between an honest client and the server. Client-privacy roughly states that no passive adversary (including the server) can infer the honest client's location after intercepting all protocol messages. Server-privacy requires that a malicious client cannot use location queries for compromising the server's database. However, Yang and Järvinen [12] recently unveiled a practical attack (which will be called as chosen fingerprint attack) for breaking the server-privacy of PriWFL. In this chosen fingerprint attack, the malicious client chooses special fingerprints, such as all-zeros or single-one fingerprints, to compromise the whole server's database. Unfortunately, their attack idea can be applied to break also the protocol recently proposed by [15], as shown in [16]. One of the major problems here is that the server-privacy defined in [13, 15] cannot cover the malicious client attack of [12]. Hence, PriWFL has not been provably demonstrated to provide security against such attack (due to lack of formal definitions). Namely, the curious-but-honest setting is not enough for proving the security for PPIL schemes.

To fix the problem of PriWFL, Yang and Järvinen proposed a new PPIL scheme (which will be referred to as YJ scheme) that relies on Paillier's public key encryption (PKE) [17] and garbled circuits based secure evaluation function (SFE). Intuitively, the YJ scheme satisfies both client- and server-privacy. However, we notice that its security is only informally justified in [12] without being analyzed under an appropriate security model. Hence, there are still open questions: (i) how many active attacks it can withstand and (ii) what the security assumptions of its build blocks and the corresponding security reductions should be. The primary motivation for this work is to develop a formal security model that allows formal analysis of the security of practical PPIL protocols.

We stress that the definitions on client- and server-privacy respectively are fundamental to the success of 'provably secure' PPIL schemes. It is therefore highly desirable to define a security model to cover the state-of-the-art attacks so that their securities can be formally proved to satisfy the security goals. Recently, Zhang et al. [15] made an effort to formulate the client- and server-privacyin a curious-but-honest setting. The definitions of client- and server-privacy in [15] can be seen as extensions from that in [13]. In the location privacy attack [15, Definition 1], a successful adversary should compromise either a client's WiFi fingerprint or location in a query. However, in practice, an adversary may violate client-privacy via learning (for instance) sensitive information about whether the client appeared at some place or its whereabouts, even without knowing its exact location or fingerprints. In particular, the definition of server-privacy is still vague in [15]. I.e., 'a certain level of accuracy' (in [15, Definition 2]) regarding ILS provided by an adversary is not clearly formalized. Specifically, there is no way to measure the accuracy of an adversary's ILS as there is no security experiment or any formulation about the adversary's advantage on breaking either client- or server-privacy. Furthermore, several important practical attacks are not modeled in [15]) such as: (i) chosen fingerprint attack introduced by Yang and Järvinen [12], (ii) known location attack (e.g. whether knowledge of an exposed (historical) location

of a client affects the client's unexposed locations), and (iii) known sub-area attacks (e.g. a follower is curious about the direction of movement or location of a client within a specific area). It is still an open question on modeling these malicious attacks. Hence, we conclude that Zhang et al.'s model is rather weak and informal and it is not possible to give a thorough security analysis for a PPIL protocol using such model.

Our results. In this paper, we present the first *unilateral-malicious* security model for WiFi fingerprint-based PPIL schemes to solve the open problems in existing models. Generally speaking, the unilateral-malicious setting is stronger than the traditional semi-honest setting but weaker than the fully malicious setting. In the unilateral-malicious setting, we particularly formulate the malicious behaviors relative to clients' sessions, e.g., manipulating WiFi fingerprints and exposing locations. We require the server to behave in semi-honest manner (for simplicity). Namely, the server may be curious about a client's location, but it should honestly run the protocol instance in order to provide a good service. We can weaken the security requirement of the server since a server's malicious behaviors (e.g., dishonest executions) would be easily caught in practice (and substantially punished) due to providing poor ILS. If the service is poor, then clients would likely just stop using the service and, consequently, make such an attack impossible. However, the server cannot easily identify a client's malicious behaviors. This is true especially when the client's messages are (non-deterministically) encrypted by its own public key. Hence, we define the first practical formal PPIL security model that focuses on modeling the most harmful malicious behaviors on the client side. We specifically apply our new security model to analyze the YJ scheme (as an example) to not only show the validity of our security model but also to exhibit another attractiveness of the YJ scheme in its provable security.

We consider the security model in a simulation environment (which covers the real world applications) with a single server and multiple clients, where each client may have multiple sessions for querying different locations. Unlike previous work [13, 15, 12], we formulate the attacks of an adversary via a series of oracle queries. Each query stands for a generic class of attacks. Under the unilateral-malicious setting, we assume that the adversary can only run protocol instances between the client and server by following the protocol specification. In spite of that, several important active attacks are defined via a series of oracle queries allowing an adversary to manipulate and learn sensitive information of sessions. Namely, an adversary can specify sessions' initial states such as WiFi fingerprint and target location area, record her own RSS measurements, or reveal a principal's long-term or ephemeral secret key and a client's location. The details of these queries can be found in Section 3.

The security goal of client-privacy is defined in an indistinguishable manner following the approach in [18]. Namely, a PPIL scheme is said to be client secure (informally) if no polynomial time adversaries can distinguish the location of an unexposed session from a random location. Whereas the security goal of server-privacy is achieved (informally) if all polynomial time adversaries are unable to generate a database D' which can provide a similar function of the server's real database D. A key problem required to be resolved is to formulate the notion of 'similar function'. Here we adopt a ball approach. Informally speaking, we say that the fabricated database D' generated by an adversary has a similar function to the real database D, if D' results in a fabricated location L' within a small ball that is centered at the corresponding real location L (which is calculated based on D for a certain location query) with a pre-defined *radius* ρ for most of the distinct location queries. Furthermore, each security goal is associated with a corresponding security experiment which defines the interactions between adversary and experiment simulator (challenger), rules of the adversary (on launching various attacks), and winning condition of the adversary. Eventually, we carefully define the client- and server- privacy in conjunction with the adversarial model, security experiment, and the corresponding adversaries' winning advantages. Here define a security model mainly for the WiFi fingerprint database. However, our security definitions and the adversarial model might be still generic enough to address the security for different kinds of PPIL schemes. It is not hard to see that the key elements (or formulation ideas) of our security model, such as

adversary model, security experiment, and security definitions, can be simply applied to formulate other types of databases with small changes.

In the security analysis of the YJ scheme, we first show that the client-privacy can be linearly reduced to that of Paillier PKE and SFE. We also show that the YJ scheme does not leak any useful information about a server's database to the adversaries due to the large enough randomness space, and the security of SFE. Since adversaries cannot gain overwhelming advantages from the messages of YJ protocol, the security of the database is therefore determined by the secret entropy of the database itself.

To test the hardness of breaking server-privacy, we deliberately design experiments for emulating the database similarity attack based on a real-world fingerprint database measured by Lohan et al.[19, BUILDING1_NEW] and a simulated one. A candidate attack is developed to automatically generate fabricated database relying on leaked fingerprints (from the target database). Via our attack, we study the possible lower bound of (unexposed) bits that adversaries have to compromise for generating a successful database. We figure out that the unexposed bits are proportional to the numbers of leaked fingerprints and similar radius ρ (meters). For example, for $\rho = 5$ and 40% of leaked fingerprints, the adversary can output a similar database with overwhelming probability. These experimental results could not only support our security analysis on server-privacy but also help practitioners to figure out ways to mitigate the database leakage threats (e.g. increasing WiFi access points).

Organization. The remainder of this paper is organized as follows. The security assumptions on the building blocks of the YJ scheme are reviewed in Section 2. In Section 3, we introduce a new security model for PPIL protocols. In Section 4, we introduce experiments for emulating a database similarity attack. In Section 5, we review the YJ scheme and introduce the security analysis under our proposed model. Finally, we give conclusion remarks in Section 6.

2 Preliminaries

General Notations. We let $\kappa \in \mathbb{N}$ be the security parameter and 1^κ be a string of κ ones. Let $[n] = \{1,\ldots,n\} \subset \mathbb{N}$ denote the set of integers. Let $a \xleftarrow{\$} S$ denote the operation sampling a uniform random element from a set S. We use $\|$ to denote the concatenation operation of two strings. Let $|\cdot|$ denote an operation calculating the bit-length of a string, and # denote an operation calculating the number of elements in a set.

Paillier Public Encryption Scheme. Paillier public-key encryption (PKE) scheme [17] is a probabilistic encryption scheme. 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 1^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 value $r \xleftarrow{\$} [n]$, and computes the ciphertext: $C = g^m \cdot r^n \bmod n^2$. 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 outputs $m = \frac{L(C^\lambda) \bmod n^2}{L(g^\lambda) \bmod n^2} \bmod n$ where $L(u) = \frac{u-1}{n}$.

Paillier PKE scheme is additively homomorphic over the group $\mathbb{Z}_n$. Namely, for two ciphertexts $C_1 = \mathsf{Enc}(pk,m_1)$ and $C_1 = \mathsf{Enc}(pk,m_2)$, we have that $\mathsf{Dec}(sk, C_1 \cdot C_2 \bmod n^2) = m_1 + m_2 \pmod n$ and $\mathsf{Dec}(sk, C_1 \cdot C_2^{-1} \bmod n^2) = m_1 - m_2 \pmod n$, where the inverse can be computed via the exponentiation $C_2^{-1} = C_2^{n^2-1} \bmod n^2$. Using the above homomorphic additions, it is also possible to compute multiplica-

tions and divisions by a scalar $t \in [n]$: $\mathsf{Dec}(sk, C_1^t \bmod n^2) = t \cdot m_1 \pmod n$ and $\mathsf{Dec}(sk, C_1^{t^{-1} \bmod n} \bmod n^2) = m_1/t \pmod n$, where $t^{-1} \bmod n$ can be computed with the Extended Euclidean Algorithm.

We review the security of Paillier PKE scheme via the following definition.

Definition 1. *The security experiment for a Paillier PKE scheme* $\mathsf{Pai}=(\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$ *is defined in the following:*

$$
\begin{array}{l}
\mathsf{EXP}^{\mathsf{ind\text{-}cpa}}_{\mathsf{Pai},\mathscr{B}}(\kappa): \\
\quad b \xleftarrow{\$} \{0,1\},\ p,q \xleftarrow{\$} \mathsf{PrimG}(\kappa/2),\ n = p \cdot q; g \leftarrow \mathbb{Z}^*_{n_2}, \\
\quad (m_0, m_1) \leftarrow \mathscr{B}(n,g),\ s.t.\ |m_0| = |m_1| \text{ and } 0 \le (m_0, m_1) < n\ ; \\
\quad r_0, r_1 \xleftarrow{\$} [n-1],\ C_0 := g^{m_0} \cdot r_0^n \mod n_2,\ C_1 := g^{m_1} \cdot r_1^n \mod n_2; \\
\quad b' \leftarrow \mathscr{B}(pk, C_b);\ \textit{if } b = b' \textit{ return 1, otherwise return 0.}
\end{array}
$$

We define the advantage of $\mathscr{B}$ *as:* $\mathsf{Adv}^{\mathsf{ind\text{-}cpa}}_{\mathsf{Pai},\mathscr{B}}(\kappa) := \left| \Pr[\mathsf{EXP}^{\mathsf{ind\text{-}cpa}}_{\mathsf{Pai},\mathscr{B}}(\kappa) = 1] - \frac{1}{2} \right|$. *We say that the Paillier PKE scheme* Pai *is secure, if for all probabilistic polynomial time (PPT) adversary* $\mathscr{B}$ *the advantage* $\mathsf{Adv}^{\mathsf{ind\text{-}cpa}}_{\mathsf{Pai},\mathscr{B}}(\kappa)$ *is a negligible function in* κ.

Two-party Secure Function Evaluation. We briefly review the formal notions regarding (circuit based) secure function evaluation (SFE) which is used by the YJ protocol. Given a public function $\hat{F}$, a classical SFE scheme allows two parties to run a protocol which results in party 1 learning only the outcome of $\hat{F}(x_1||x_2)$, while party 2 learning nothing, where x_1 and x_2 are the private inputs of party 1 and party 2 respectively. We refer the reader to [20] for more details on the security notions and concrete example of SFE.

We let $\hat{f}$ denote a circuit for a certain function $\hat{F}$ with input size $n \in \mathbb{N}$ (that may be accessed as $\hat{f}.n$). And let $\mathsf{ev}(\hat{f}, x)$ be a canonical circuit evaluation function which takes as inputs $\hat{f}$ and a string x, and computes the output of the function $\hat{F}(x)$. Here we define a function $\Phi(\hat{f})$ to describe what we allow to be revealed regarding $\hat{f}$. With respect to a garbling scheme, Φ may reveal a circuit's size, topology, identity, or many others. More concrete side information functions can be found in [20, 21].

In a two-party protocol, we suppose that party i ($i \in [2]$) has a private string x_i with length n_i, and party 2 has a circuit $\hat{f}$ where $n = n_1 + n_2$. We describe a two-party protocol (for executing a SFE scheme) via a pair of PPT algorithms $\Sigma = (\Sigma_1, \Sigma_2)$. Party $i \in \{1,2\}$ will run Σ_i on its current state and the incoming message from its intended partner, to generate an outgoing message and a local output. The initial state of Σ_i includes the security parameter κ, a fresh random coin $\gamma_i \xleftarrow{\$} \mathscr{R}_i$ (chosen from a random space $\mathscr{R}_i$) and the (private) function input I_i of party i. The random coins γ_1 and γ_2 might be omitted (in the following descriptions) for simplicity, i.e., they are implicitly generated and used. In order to represent the protocol execution, we define a PPT algorithm View^i_Σ which takes as input security parameter 1^κ, and inputs (I_1, I_2) for the two parties respectively, and returns an execution view vw_i and output out_i of party i in a protocol instance. Nevertheless, we may denote an execution between two parties as $\mathsf{SF}.\Sigma(I_1, I_2)$ at a high-level view.

Then a SFE scheme is a tuple $\mathsf{SF} = (\Sigma, \mathsf{ev})$ where Σ is a two-party protocol with input (I_1, I_2) as above and ev is a circuit evaluation function. The correctness requirement states that, for all $\hat{f}$ and all $x \in \{0,1\}^{\hat{f}.n}$, we have $\Pr[\mathsf{out}_1 = \mathsf{ev}(\hat{f}, x)] = 1$, where $x = x_1||x_2$, $x_1 \in I_1$ and $(x_2, \hat{f}) \in I_2$. We here review the privacy of SFE in the honest-but-curious setting.

Definition 2. *For a SFE scheme* $\mathsf{SF} = (\Sigma, \mathsf{ev})$*, a simulator* $\mathscr{S}$ *and an adversary* $\mathscr{E}$*, the security experiment relative to* Φ *is defined as follows:*

$$
\begin{array}{l|l}
\mathsf{EXP}^{\mathsf{pri.sim},\mathscr{S}}_{\mathsf{SF},\mathscr{E},\Phi}(\kappa, i): & \mathsf{Excute}_{\mathsf{SF}}(b, i, x_i, \hat{f}): \\
\quad b \xleftarrow{\$} \{0,1\}; & \quad \textit{if } x_i \not\subseteq \{0,1\}^{\hat{f}.n_i} \textit{ return } \perp; \\
\quad b' \leftarrow \mathscr{E}^{\mathsf{Excute}_{\mathsf{SF}}(b,i,\cdot,\cdot)}(\kappa, i); & \quad x_{3-i} \xleftarrow{\$} \{0,1\}^{\hat{f}.n_{3-i}},\ I_1 := x_1,\ I_2 := (x_2, \hat{f}); \\
\quad \textit{if } b = b' \textit{ return 1,} & \quad \textit{if } b = 1 \textit{ return } \mathsf{View}^i_\Sigma(1^\kappa, I_1, I_2); \\
\quad \textit{otherwise return 0.} & \quad \textit{if } i = 1 \textit{ return } \mathscr{S}(1^\kappa, \mathsf{ev}(\hat{f}, x_1||x_2), \Phi(\hat{f})); \\
 & \quad \textit{if } i = 2, \textit{ return } \mathscr{S}(1^\kappa, \hat{f}, |x_1|);
\end{array}
$$

We define the advantage of $\mathscr{E}$, which is allowed only a single $\mathsf{Excute}_{\mathsf{SF}}$ *query, in the above experiment as:* $\mathsf{Adv}^{\mathsf{pri.sim},\mathscr{S}}_{\mathsf{SF},\mathscr{E},\Phi}(\kappa,i) := \left|\Pr[\mathsf{EXP}^{\mathsf{pri.sim},\mathscr{S}}_{\mathsf{SF},\mathscr{E},\Phi}(\kappa,i)=1]-\frac{1}{2}\right|$. *We say that* SF *is secure relative to* Φ*, if for each* $i \in \{1,2\}$ *and all PPT adversaries* $\mathscr{E}$*, the advantage* $\mathsf{Adv}^{\mathsf{pri.sim},\mathscr{S}}_{\mathsf{SF},\mathscr{E},\Phi}(\kappa,i)$ *is a negligible function in* κ.

3 A New Security Model for Privacy Preserving Indoor Location Schemes

In this section, we define a new unilateral-malicious security model for privacy preserving indoor location (PPIL) protocols which are based on WiFi fingerprints. The privacy for client and server is formulated respectively following the well-known game-based modeling approach [18, 22].

Simulation Preliminary. We first describe the general simulation environment which will be exploited in the following security notions (in particular for security experiment). There are two types of entities considered: client C and server S. The server S is supposed to provide the indoor location service (ILS) of a building according to a client's request. The building area (which is covered by the location service) is assumed to be delicately divided into M reference locations $\mathsf{LT} = \{i,(x_i,y_i,z_i)\}_{i=1}^{M}$, e.g. the black dot in Figure 1, where (x_i,y_i) denotes the horizontal coordinates and z_i denotes the vertical coordinate (e.g., the position of a floor). One could consider the unit of each coordinate is meter (m) for instance. Moreover, the building is deployed with N WiFi access points (AP) to provide network service, where each i-th ($i \in [N]$) access point may have a unique identity AP_i. Let $\mathsf{APT} = \{AP_j\}_{j=1}^{N}$ be list storing all identities of WiFi access points. In particular, each location has a so-called WiFi fingerprint which comprises of Received Signal Strength (RSS) values of certain WiFi AP, where each RSS value is from a range $\mathscr{R}_v = [v_{min}, v_{max}]$ and (v_{min}, v_{max}) are minimum and maximum values respectively. Consequently, the server is assumed to hold a pre-measured WiFi fingerprint database D which consists of a set of tuples $< i, V_i = \{v_{i,j}\}_{j=1}^{N} >_{i=1}^{M}$ (See also in [12, Table 1]) , where i is an index of a reference location $L_i \in \mathsf{LT}$, each $v_{i,j}$ denotes the RSS value obtained at L_i from AP_j. Furthermore, we let Dist be a distance function which takes as input two locations L_i and L_j (with their corresponding coordinates (x_i,y_i,z_i) and (x_j,y_j,z_j)) and outputs the distance between them. One could consider Euclidean distance, i.e. equation 1, as a concrete example of Dist.

When C wants to know its location, it first measures the RSS values from all APs to get a real-time WiFi fingerprint $F = \{f_j\}_{j=1}^{N}$. Then it may 'privately' submit F to S as a location query, and calculate its location L from S's response. We refer the reader to [12, §2.1] for more details on the principle of WiFi fingerprint localization. Meanwhile, the private information of the client mainly includes its secret key sk, location query F and the corresponding location L. The secret of the server is the database D.

In order to emulate the behaviors of a set of entities (including λ clients and 1 server), we may realize a collection of oracles $\{\pi_\tau^s, \pi_{\lambda+1}^t : \tau \in [\lambda], s \in [d], t \in [\lambda \times d]\}$ for $(\lambda,d) \in \mathbb{N}$. Each oracle π_τ^s behaves as the s-th protocol instance (session) performed by the party τ for calculating one location. The special party $\lambda+1$ is assumed to be server. Each party may have a pair of pubic/private key (pk_τ, sk_τ) for $\tau \in [\lambda+1]$, where pk_τ can be accessed by all oracles. Moreover, each oracle π_τ^s for $\tau \in [\lambda]$ is supposed to keep the following internal state variables: (i) $ds_\tau^s \in \{\mathsf{accept}, \mathsf{reject}\}$ – final decision of a session; (ii) F_τ^s – fingerprint $F_\tau^s = \{v'_j\}_{j=1}^{N}$ for a location query; (iii) ins_τ^s – index selection set (INS) specifying the location indexes (in LT) which are close to the current location related to F_τ^s; (iv) er_τ^s – ephemeral randomness used to run the protocol instance; (v) T_τ^s – transcript recording all sent and received protocol messages; (vi) $L_\tau^s = (x_\tau^s, y_\tau^s, z_\tau^s)$ – location of party τ calculated in the s-th session. We assume the variable L_τ^s will be assigned if and only if $ds_\tau^s = \mathsf{accept}$ (meaning that a protocol instance is correctly executed in a session). The server's oracles only have ds_τ^s and T_τ^s.

In order to simulate a WiFi fingerprint used by a location query, we define a function $\mathsf{FPTSim}(i)$ which on input a reference location index i generates a WiFi fingerprint $F_i = \{f_j\}_{j=1}^{N}$ with the following

steps: (i) $f_j \xleftarrow{\$} [v_{i,j}-\Delta, v_{i,j}+\Delta]$ where Δ is a pre-defined positive integer, where $v_{i,j} \in \mathsf{D}$; (ii) If $f_j \leq v_{min}$ or $v_{i,j} = v_{min}$ then $f_j := v_{min}$; (iii) Else if $f_j \geq v_{max}$ then $f_j := v_{max}$.

Adversarial Model. Here we define the power of an active adversaries. The active adversaries $\mathscr{A}$ in our model are considered as a probabilistic polynomial time (PPT) algorithms, which may interact with another PPT algorithm called simulator $\mathscr{C}$ via the following queries:

- **InitCorruptO**$(\tau, s, \tilde{F})$: The variables ds^s_τ, T^s_τ and L^s_τ (if any) of the client's oracle π^s_τ are initiated with an empty string $\emptyset$. This query initializes $\mathsf{ins}^s_\tau := [M]$. If $\tilde{F} \neq \emptyset$ and $\tau \neq \lambda+1$, this query sets $F^s_\tau := \tilde{F}$. Each oracle can be initialized by this query only once.
- **InitHonestO**(τ, s, i, rds): This query first initializes ds^s_τ, er^s_τ, T^s_τ and L^s_τ with empty string $\emptyset$. Let $\widetilde{\mathsf{ins}} \subseteq [M]$ be a set of location indexes such that $\forall j \in \widetilde{\mathsf{ins}}$ the distance between $L_i = (x_i, y_i, z_i)$ and $L_j = (x_j, y_j, z_j)$ is smaller than rds, i.e., $\mathsf{Dist}(L_i, L_j) \leq rds$. Note that $\widetilde{\mathsf{ins}}$ may cover indexes within a ball centered at i with radius rds. If $\tau \neq \lambda+1$ and $\#\widetilde{\mathsf{ins}} \geq \lceil \chi \cdot M \rceil$ (for a threshold say $0.1 \leq \chi \leq 1$)[1], this query initializes F^s_τ as follows: (i) $j \xleftarrow{\$} \widetilde{\mathsf{ins}}$; (ii) $F^s_\tau := \mathsf{FPTSim}(j)$; (iii) $\mathsf{ins}^s_\tau := \widetilde{\mathsf{ins}}$. Again each client's oracle can be initialized by this query only once.
- **Execute**$_{\mathsf{PPIL}}(\tau, s, t)$: This query executes the protocol instance between an unused and initialized client's oracle π^s_τ and a server's unused oracle $\pi^t_{\lambda+1}$, and returns the protocol transcript T^s_τ. We call π^s_τ and $\pi^t_{\lambda+1}$ proceeded in this query as *partner oracles*. The oracles run by this query are called *used*. All server's oracles here are assumed to be default initialized (without specific initiation query).
- **CorruptC**(τ): This query responds with the τ-th client's secret key sk_τ.
- **CorruptS**: This query responds with the server's database D and secret key $sk_{\lambda+1}$ (if any).
- **RandReveal**(τ, s): Oracle π^s_τ responds with the ephemeral secret key er^s_τ.
- **LocReveal**(τ, s): Oracle π^s_τ responds with the location L^s_τ.
- **LocTest**(τ, s): If the oracle has state $ds^s_\tau \neq \mathsf{accept}$ or $\tau = \lambda+1$, then this query returns a failure symbol $\perp$. Otherwise, it does the following steps: (i) flip a fair coin $b \xleftarrow{\$} \{0,1\}$; (ii) choose a random index $j \in \mathsf{ins}^s_\tau$, obtain $F_j := \mathsf{FPTSim}(j)$, and calculate L_0 based on F_j and D (following the protocol specification) such that $L_0 \neq L^s_\tau$; (iii) set $L_1 := L^s_\tau$ (which is the real location). Eventually, the location L_b is returned. This query is allowed to be asked at most once during the following corresponding security experiment. We call the oracle π^s_τ selected in this query as *test oracle*.
- **DBLeak**(i): If the index i has been queried via this query, then it returns a failure symbol $\perp$. Otherwise, this query responses with a similar WiFi fingerprint $F'_i \leftarrow \mathsf{FPTSim}(i)$ according to the i-th row of database D.

InitCorruptO query is used to model the chosen fingerprint attacks against server's privacy (in the unilateral-malicious setting), i.e., the malicious client may choose special fingerprints (e.g. all zeros) to compromise the server's database. For example, the attack introduced in [12] is a kind of chosen fingerprint attack. An oracle initialized by this query is known as location exposed oracle.

InitHonestO query is used to initialize the honest (unexposed) oracle based on an area which is specified by an adversary in term of the reference location index i and a radius rds. We categorize the attacks modeled by this query as *known sub-area attacks*. Consider the attack scenario that an adversary loses his tracking target at a street corner (determined by i) and he wants to know the target's 'whereabouts' (within a range rds). In this case, the attacker may know an approximate area of the client within a range. Moreover, if rds is large enough then it may cover all location indexes in LT.

Execute$_{\mathsf{PPIL}}$ query formulates the passive adversaries which only observe the communication between the client and server.

CorruptC and **CorruptS** queries formulate the corruption of an honest principal's long-term cre-

[1] If χ is too small, then there is no privacy at all.

dentials respectively. The corrupted party is known as dishonest or malicious one.

RandReveal query models the randomness exposure attacks which may be caused by malware or careless disposal.

DBLeak query 'approximately' formulates the attack that $\mathscr{A}$ measures and records the WiFi fingerprints V_i' (which is similar to V_i of D) for certain location index i, say based on limited WiFi fingerprint samples.

LocReveal query models the known location attacks (ULA). The resilience of ULA requires that the exposed locations will not affect the others. For example, the PPIL scheme proposed in [14] is subject to known location attack. To get a location, the client in [14] would issue a set of camouflaged localization requests that follow a similar natural movement pattern. However, if one of the client's locations is exposed, e.g., by posting a picture, then the server can simply identify which location request is the real one.

LocTest query will be exploited to formulate the capability of an adversary on breaking the client's privacy. The job of the adversary is to distinguish the bit chosen by the **LocTest** query.

Note that we are the first one to generalize the practical attacks against PPIL schemes via the above generic queries which have not been formalized in previous work [13, 15, 12].

Client Privacy. We first define a security experiment as follows.

SECURITY EXPERIMENT $\mathsf{EXP}^{\mathsf{CP}}_{\Pi,\mathscr{A}}(\kappa,\mathsf{D})$: On input security parameter κ and a server's database D, the security experiment is carried out as a game between a simulator $\mathscr{C}$ and an adversary $\mathscr{A}$ based on a PPIL scheme Π, where the following steps are performed:

1. The simulator $\mathscr{C}$ first initiates the game by realizing a collection of oracles and generating all public/private key pairs for all $\lambda+1$ honest parties and all other public information. $\mathscr{C}$ gives $\mathscr{A}$ all public information $\{pk_\tau\}_{\tau=1}^{\lambda+1}$, LT, APT and $\mathscr{PD}$.

2. $\mathscr{A}$ may adaptively issue a polynomial number of **InitCorruptO**, **InitHonestO**, **Execute**$_{\mathsf{PPIL}}$, **CorruptC**, **CorruptS**, **LocReveal**, and **RandReveal** queries. At some point, $\mathscr{A}$ may issue a single **LocTest**(τ^*,s^*) query.

3. At the end of the game, $\mathscr{A}$ may terminate and output a bit b' as its guess for b of **LocTest**(τ^*,s^*) query.

4. Meanwhile, the experiment would return a failure symbol $\perp$ if one of the following conditions is satisfied: (a) $\mathscr{A}$ has not issued a **LocTest**(τ^*,s^*) query; (b) The **LocTest**(τ^*,s) query returns a failure symbol $\perp$; (c) $\mathscr{A}$ asked an **InitCorruptO**(τ^*,s^*,F^*) query to the test oracle; (d) $\mathscr{A}$ asked a **CorruptC**(τ^*) query; (e) $\mathscr{A}$ asked either a **RandReveal**(τ^*,s^*) query or a **RandReveal**$(\lambda+1,t^*)$ query, where $\pi_{\lambda+1}^{t^*}$ is the partner oracle of the test oracle; (f) $\mathscr{A}$ asked a **LocReveal**(τ^*,s^*) query to the test oracle $\pi_{\tau^*}^{s^*}$.

5. The experiment finally returns 1 if $b=b'$, and 0 otherwise.

We call an adversary as a 'legal' one if it runs an experiment without failure. A legal adversary should not violate the rules defined in the above step 4). Note that violating one of the rules c) to f) would 'trivially' break the client-privacy, i.e., asking the corresponding queries (specified in the rules) would enable the adversary to easily distinguish the bit b chosen in the **LocTest**(τ^*,s) query without breaking the underlying protocol. These situations should be therefore forbidden in the experiment. Otherwise, it would always return 1.

Definition 3 (Client-privacy). *The advantage of legal adversaries $\mathscr{A}$ in the above experiment is* $\mathsf{Adv}^{\mathsf{CP}}_{\Pi,\mathscr{A}}(\kappa,\mathsf{D}) := \left|\Pr[\mathsf{EXP}^{\mathsf{CP}}_{\Pi,\mathscr{A}}(\kappa,\mathsf{D})=1]-\frac{1}{2}\right|$. *We say that a PPIL scheme Π is client-secure, if for all PPT legal adversaries $\mathscr{A}$, the advantage* $\mathsf{Adv}^{\mathsf{CP}}_{\Pi,\mathscr{A}}(\kappa,\mathsf{D})$ *is a negligible function in* κ.

Server Privacy. Informally speaking, the server's privacy is achieved if all polynomial time adversaries are unable to generate a database D' which can provide a similar function as the server's real database D. We may call a location calculated based on D' as a *fabricated location*, and a location calculated based on D as *real location*. Given two databases D and D', we have the following similar event (as exemplified in Figure 1):

- *Similar Event (SE)*: For a client's location query regarding WiFi fingerprint $F_i = \{f_j\}_{j=1}^{N}$, the corresponding location L_i and the fabricated location L_i' have distance at most ρ, i.e., $\mathsf{Dist}(L_i, L_i') \leq \rho$, where ρ is a pre-defined difference threshold (in meter).

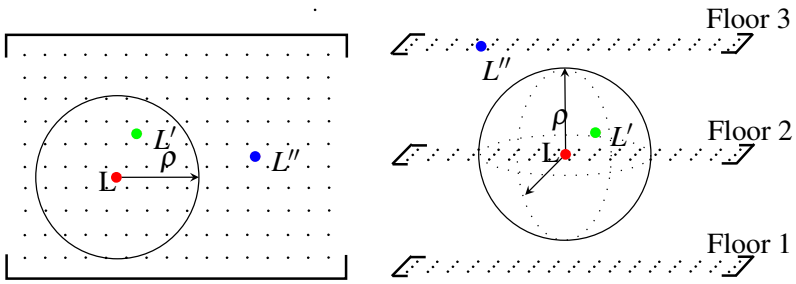


Figure 1: Similar event occurrence examples in horizontal (left) and vertical planes (right). The small black dots represent the reference locations in LT. The red dot represents the real location L. The green dot represents the fabricated location L' in which the similar event occurs. The blue dot represents the fabricated location L'' in which the similar event does not occur.

The term on 'similar function' of two databases can be roughly illustrated as follows. Given a number of distinct client's location queries, the occurrence rate of SE is larger than a pre-defined success threshold α (e.g. $\alpha = 0.7$). Let TF be a test set that consists of $|TF| > M$ distinct fingerprints of random locations. For example, one could generate a fingerprint $F \in TF$ by randomly choosing an index $i \xleftarrow{\$} [M]$ and running $F := \mathsf{FPTSim}(i)$. Let $\mathsf{SimilarTest}$ be a function that is used to test the functional similarity between two databases. $\mathsf{SimilarTest}$ takes as input two databases D and D' with their related reference location lists LT and LT' (respectively), and a test set TF, and outputs the test result in $\{0,1\}$. The execution steps of $\mathsf{SimilarTest}$ comprises of the following:

- Initiate a SE count variable $cnt := 0$. Suppose that for a fingerprint $F_i \in TF$ the real location which is calculated based on F_i, D and LT is $L_i = (x_i, y_i, z_i)$, and the fabricated location which is calculated based on F_i, D' and LT' is $L_i' = (x_i', y_i', z_i')$. For $i \in [|TF|]$, if $\mathsf{Dist}(L_i, L_i') \leq \rho$ then $cnt := cnt + 1$.
- Finally, it returns 1 if $\frac{cnt}{|TF|} > \alpha$; otherwise, 0 is returned.

The parameters, which are relevant to the formulation of the server-privacy, are summarized in Table 1.

SECURITY EXPERIMENT $\mathsf{EXP}^{\mathsf{SP}}_{\Pi,\mathscr{A}}(\kappa,\mathsf{D},\mathsf{LT},\rho,\alpha,\phi)$: On input security parameter κ, a server's database D, and a distance accuracy threshold ρ, the security experiment is carried out as a game between a simulator $\mathscr{C}$ and an adversary $\mathscr{A}$ based on a PPIL scheme Π, where the following steps are performed:

1. The simulator $\mathscr{C}$ first implements a collection of oracles and generates all public/private key pairs for all $\lambda + 1$ honest parties and all other public information. All public information are given to $\mathscr{A}$.

Params	Description
D	real database of server
ϕ	a security parameter specifying the number of **DBLeak** queries
ρ	distance threshold between the real location and the fabricated location
α	probability threshold of SE
TF	test set of random fingerprints

Table 1: Parameters of server-privacy.

2. $\mathscr{A}$ may issue a polynomial number of queries to **InitCorruptO**, **CorruptC**, **Execute**$_{\mathsf{PPIL}}$, **RandReveal**, and **LocReveal** respectively, and at most ϕ **DBLeak** queries.

3. Eventually, $\mathscr{A}$ may return a database D' and a relevant reference location list LT' that has M' reference location. Meanwhile, the experiment would return a failure symbol $\perp$ if $\mathscr{A}$ asked either a **RandReveal**$(\lambda+1,\cdot)$ query or more than ϕ queries to **DBLeak**.

4. Finally, the experiment returns $\mathsf{SimilarTest}(\mathsf{D},\mathsf{D}',\mathsf{LT},\mathsf{LT}',TF)$.

Definition 4 (Server-privacy)**.** *The advantage of a legal adversary $\mathscr{A}$ in the above experiment is* $\mathsf{Adv}^{\mathsf{SP}}_{\Pi,\mathscr{A}}(\kappa,\mathsf{D},\mathsf{LT},\rho,\alpha,\phi) := \Pr[\mathsf{EXP}^{\mathsf{SP}}_{\Pi,\mathscr{A}}(\kappa,\mathsf{D},\mathsf{LT},\rho,\alpha,\phi)=1]$. *We say that a PPIL scheme Π is server-secure, if for all PPT legal adversaries $\mathscr{A}$, the advantage* $\mathsf{Adv}^{\mathsf{SP}}_{\Pi,\mathscr{A}}(\kappa,\mathsf{D},\mathsf{LT},\rho,\alpha,\phi)$ *is a negligible function in κ.*

We define the above model based on WiFi fingerprint database as an example. Of course, one could simply modify our model for other types of PPIL schemes since each query aforementioned represents a generic class of attacks against PPIL schemes. One may only need to customize the simulation environment and slightly modify the queries if necessary.

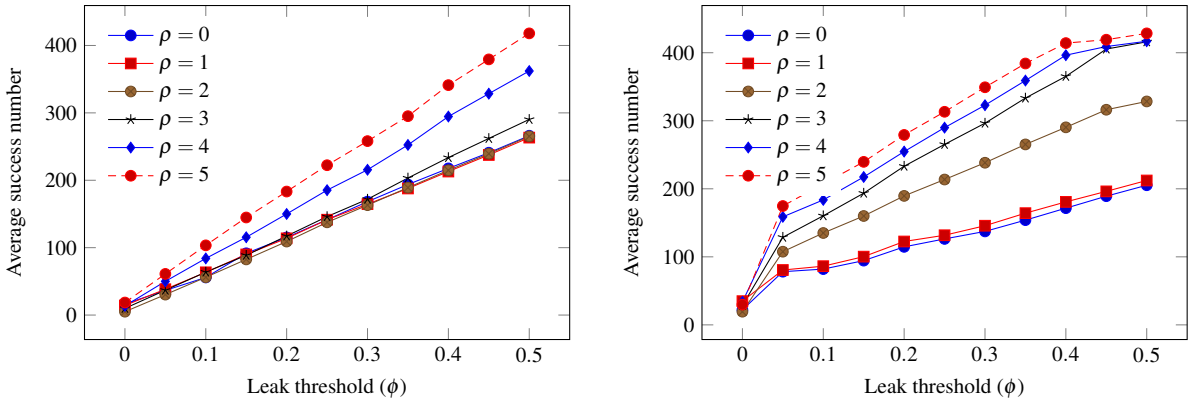
Database Hardcore. The volume of a database D is determined by the number M of reference locations (that is related to the area of a building), the number N of APs, and bit size of each RSS value $|\mathscr{R}_v|$. However, there is a general problem on how hard it is for adversaries to generate a valid fabricated database D' without any useful information from a PPIL scheme using D. I.e. is the D' itself hard to build? This question is independent of any concrete PPIL schemes. If D' is easy to generate without breaking the PPIL scheme, then we do not need a PPIL scheme at all. Since the server could just publish its database for all clients. Intuitively, the adversary should be very hard to generate a valid fabricated D' that has a similar function as D since D' also has a large number of bits to predict. In the following, we are going to give a formal definition regarding the security assumption of a database (that is non-relevant to PPIL schemes).

Definition 5. *The security experiment for testing the hardness of forging a similar database for a target database D is defined in the following:*

$$\mathsf{EXP}^{\mathsf{DBH}}_{\mathscr{D}}(\kappa,\mathsf{D},\mathsf{LT},\rho,\alpha,\phi):$$
$$(D',\mathsf{LT}') \leftarrow \mathscr{D}^{\textbf{\textit{DBLeak}}(\cdot)}(\mathsf{LT},\rho,\alpha,\phi),\ \textit{Return}\ \mathsf{SimilarTest}(\mathsf{D},\mathsf{D}',\mathsf{LT},\mathsf{LT}',\rho,\alpha,\phi).$$

*The advantage of $\mathscr{D}$ which can ask at most ϕ **DBLeak** queries in the above experiment is* $\mathsf{Adv}^{\mathsf{DBH}}_{\mathscr{D}}(\kappa,\mathsf{D},\mathsf{LT},\rho,\alpha,\phi) := \Pr[\mathsf{EXP}^{\mathsf{DBH}}_{\mathscr{D}}(\kappa,\mathsf{D},\mathsf{LT},\rho,\alpha,\phi)=1]$. *We say that a database D is hard to forge, if for all PPT adversaries $\mathscr{D}$ the advantage* $\mathsf{Adv}^{\mathsf{DBH}}_{\mathscr{D}}(\kappa,\mathsf{D},\mathsf{LT},\rho,\alpha,\phi)$ *is a negligible function in κ.*

It is straightforward to see that D is hard to forge if only a small portion of D is leaked via **DBLeak** to the adversary and D has large M, N, and $|\mathscr{R}_v|$, e.g., $M=505$, $N=241$ and $|\mathscr{R}_v|=8$ in the real database



(a) Against real database (b) Against simulated database

Figure 2: Average success number of distinct location queries based on fabricated database. Distances are compared to real database (a) and simulated database (b).

[19, BUILDING1_NEW] which has $M \times N \times |\mathscr{R}_v| = 973640$ bits at all. However, an open question is how hard it is to create a valid fabricated database. Such hardness might be closely related to the structure of specific building and database generation algorithm. In the future work, one is encouraged to formally analyze the database hardcore assumption in the setting with the leakage of side-channel information, such as adversaries' own RSS measurements modeled by **DBLeak** query. In this paper, we just focus on the formalism of server-privacy for PPIL schemes.

Leak threshold ϕ	Similar threshold ρ						Leak threshold ϕ	Similar threshold $\rho = 5$
	0	1	2	3	4	5		
0.5	0	0	0	0.01	0.86	1	0.42	0.962
0.45	0	0	0	0	0.05	1	0.41	0.785
0.4	0	0	0	0	0	0.38	0.39	0.185
0.3	0	0	0	0	0	0	0.38	0
0.1	0	0	0	0	0	0	0.37	0
0	0	0	0	0	0	0	0.36	0

Table 2: Success probability for $\alpha = 0.7$ against real database in 100 tests (left) and in 1000 tests (right) respectively.

Leak threshold ϕ	Similar threshold ρ						Leak threshold ϕ	Similar threshold $\rho = 3$
	0	1	2	3	4	5		
0.5	0	0	0.03	1	1	1	0.36	0.211
0.45	0	0	0	1	1	1	0.35	0.032
0.4	0	0	0	0.73	1	1	0.34	0
0.3	0	0	0	0	0	0.33	0.33	0
0.1	0	0	0	0	0	0	0.32	0
0	0	0	0	0	0	0	0.31	0

Table 3: Success probability for $\alpha = 0.7$ against simulated database in 100 tests (left) and in 1000 tests (right) respectively.

4 Database Hardcore Analysis

In this section, we study the problem concerning: how hard is for an adversary to generate a similar database (i.e., satisfying $\Pr[SE] > \alpha$). We investigate this problem in presence of adversaries which may measure the fingerprints at some location herself. This implies that an adversary may have some prior knowledge about the target database D. We are going to show if the number of leaked reference fingerprints from D is not small, it might be hard to create a valid fabricate database.

We carry out our experiments based on two types of database: (i) a real WiFi fingerprint database [19, BUILDING1_NEW], and (ii) a simulated database. There are 505 reference locations and 241 WiFi access points in both databases. The first fingerprint of the simulated test database is generated randomly with a zero threshold around 0.8. All other fingerprints in the simulated database are generated relying on its adjacent one with a random derivation (e.g. ranging from 1 to 8). Hence, the simulated database has many co-related fingerprints.

In order to generate a fabricated database (fD), we design an 'automatic' database similarity attack (DSA). In this attack, we mainly make use of the leaked fingerprint from the real database (D) and a guessing strategy in terms of the similarity of fingerprints to generate fD. Let μ be a similarity constant specifying how many fingerprints can be derived from a leaked fingerprint (via **DBLeak** query) in a similarity attack. It is not hard to see that the value of μ is determined by the structure of a database D. We can calculate μ based on the average number of each reference fingerprint's closed (similar) fingerprints (e.g. in the sense of Euclidean distance). For the real database [19, BUILDING1_NEW], we have $\mu = 2$. Let ϕ be the leak threshold specifying the ratio of **DBLeak** queries, and $q_c = \lfloor \phi \cdot M \rfloor$.

In the following, we roughly illustrate our attack steps for $\mu = 2$:

- Ask q_c **DBLeak**(i) queries to even indexes i ($0 \le i \le 2 \cdot q_c$), where $q_c \le \lfloor \frac{M}{2} \rfloor$. For $0 \le i \le 2 \cdot q_c$, set the i-th fingerprint in fabricated database as $\mathsf{fD}[i] := \mathsf{D}[i]$.
- For all odd indexes j ($1 \le j \le 2 \cdot q_c + 1$), find two references indexes (i.e., rj and lj) which are non-zero and near to j (in terms of the coordinates in LT). Then calculate the fingerprint $\mathsf{fD}[j]$ by an average of these two references indexes' fingerprints as $\mathsf{fD}[j] = (\mathsf{fD}[rj] + \mathsf{fD}[lj])/2$.
- For all other indexes t, search for a reference index j within a radius (e.g. 10 meters) centered by t. The $\mathsf{fD}[t]$ is generated in terms of the following two disjoint cases:
  - When j is found. For $1 \le \iota \le N$, if $\mathsf{fD}[j][\iota] = 0$ then $\mathsf{fD}[t][\iota] = 0$; otherwise generate a random value $\mathsf{fD}[t][\iota] \xleftarrow{\$} [\mathsf{fD}[j][\iota] - RSS_{dev}, \mathsf{fD}[j][\iota] + RSS_{dev}] (\in [v_{min}, v_{max}])$ where the constant RSS_{dev} (e.g. $RSS_{dev} = \Delta$) represents the expecting difference between these two fingerprints.
  - When j is not found. For $1 \le \iota \le N$, sample a random float number $r_\iota \xleftarrow{\$} [0,1]$, if $r_\iota > 0.8$ (where 0.8 is an approximate ratio of zeros in D)[2], then $\mathsf{fD}[t][\iota] = 0$; otherwise generate a random value $\mathsf{fD}[t][\iota] \xleftarrow{\$} [v_{min}, v_{max}]$.

The ideal cases of rj and lj in our attack strategy are expected to be $j+1$ and $j-1$. This is also why we first choose one interval to disclose the fingerprints of even indexes. Such attack scenario is based on the idea that adjacent location indexes may have very close fingerprints. Hence, the impact of leaked fingerprints could be proliferated to other unexposed location indexes as many as possible.

In the real world, attackers may have better attack solution. We here just show some reference attack results via the above attack. In particular, we carry out a number of experiments by gradually changing the parameters, i.e., leak threshold ϕ, similar threshold ρ, and success threshold α, to show the effects of them in our attack. We first do 100 tests for each different set of parameters to obtain general results.

[2]The rate of zeros can be roughly counted in terms of leaked fingerprints.

After this, we did 1000 tests for specific significant parameters to obtain more fine results. Meanwhile, we choose the nearest location index (in LT) obtained in a location query to check the similar event for simplicity, i.e., $k = 1$ in the K-NN algorithm.

From Table 2, an adversary could have an overwhelming probability around the leak threshold $\phi = 0.4$ when $\rho \leq 5$ (which may be a tolerable difference). In other words, when $\phi > 0.35$, an adversary (with better attack algorithms) might be very likely to output a similar database. Figure 2 shows the average success numbers for different sets of parameters. One can see that the success number is linear in leak threshold for both types of target databases. For $\phi = 0.35$ and $\rho = 5$, the average success number against the real database is about 300 that is very close to the boundary of success condition for $\alpha = 0.7$, i.e., $\alpha \cdot M = 0.7 \cdot 505 = 353$. However, for a highly inner co-related database (i.e. our simulated database), the attack is much easier as shown in Table 3 and Figure 2 (b). Nevertheless, under resealable thresholds $\phi = 0.3$ and $\rho \leq 3$, the database similarity attack might be very hard as well.

In terms of our experimental results, one could find out that the success probability of an adversary is closely related to the similarity (distribution) of reference fingerprints that may be determined by the mounted positions of APs. Hence, we suggest to carefully select the locations where APs are installed to reduce the co-related fingerprints. Besides the form of the target database (which is determined by a specific building), another factor might affect server-privacy is the number of non-zero RSS values in each fingerprint. Reducing the zeros in a database might be useful to enhance server-privacy. One could make use of our above attack scenario to test the security of the resultant database.

Remark 1. *Here we just give some ideas and example experimental results for testing database hardcore. In practice, adversaries may use different parameters (e.g. N and M) to create its own database. We encourage researchers to develop better experiments to figure out more accurate results. The primary goal of this paper is to formalize the server-privacy.*

5 On the Security of the YJ Scheme

The YJ Scheme. We first review the PPIL scheme [12] recently proposed by Yang and Järvinen. The YJ scheme is built from Paillier PKE $\mathsf{Pai} = (\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$ and two-party SFE protocol $\mathsf{SF} = (\Sigma, \mathsf{ev})$. Paillier PKE scheme is used to protect a client C's fingerprint $F = (f_1, f_2, \ldots, f_N)$. In the YJ scheme, the server S should compute the distances between F and V_i (of its database D), where each distance d_i is assumed to be the following Euclidean distance:

$$d_i = ||V_i - F||^2 = \sum_{j=1}^{N} (v_{i,j} - f_j)^2 = \sum_{j=1}^{N} v_{i,j}^2 + \sum_{j=1}^{N} (-2 v_{i,j} f_j) + \sum_{j=1}^{N} f_j^2. \quad (1)$$

SFE protocol is used to privately compute the location $L_{\mathsf{C}} = (x, y, z)$ of C as the centroid of the k nearest reference locations indexed by $i_1, i_2, \ldots, i_k$, where $i_1, i_2, \ldots, i_k$ indicate distances such that $d_{i_1} \leq d_{i_2} \leq \ldots \leq d_{i_k} \leq d_j$ for all $j \neq i_1, i_2, \ldots, i_k$.

PROTOCOL DESCRIPTION. When C subscribes to the location service, it runs $(sk, pk) \xleftarrow{\$} \mathsf{KeyGen}(\kappa)$ to generate a key pair (sk, pk) for Paillier PKE scheme with a sufficiently large κ (e.g. $\kappa = 2048$) and sends $pk = (n, g)$ to S. The protocol execution is shown in Figure 3.

Note that the randomness space $\mathscr{R}_R = \mathbb{Z}_n$ may result in the blinded distance being wraparound over $\mathbb{Z}_n$, i.e. a modular n operation is involved in the generation of the blinded distance.

Security Analysis. The security results of our scheme are shown by the following theorems. Here we briefly analyze the theorems.

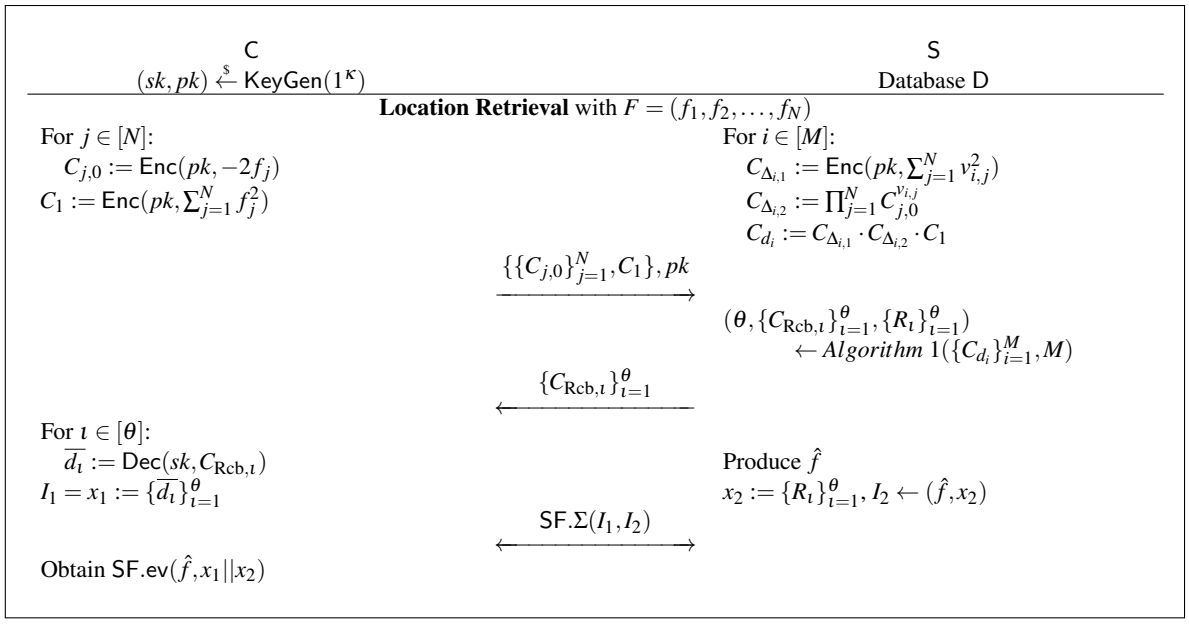


Figure 3: The YJ Scheme

Algorithm 1: Pack Encrypted Distance Set

Input: $\{C_{d_i}\}_{i=1}^M$ and M
Output: θ, $\{C_{\mathrm{Rcb},\iota}\}_{\iota=1}^\theta$, and $\{R_\iota\}_{\iota=1}^\theta$

$\theta := 1$; $\mu := M$; $\mathscr{R}_R = \mathbb{Z}_n$
while $\mu > 0$ **do**
  $t := \frac{\kappa-1}{m}$
  if $t > \mu$ **then**
    $t := \mu$
  $C_{\mathrm{cb},\theta} := \prod_{i=1}^t C_{d_{\mu-i}}^{2^{(i-1)m}}$; $R_\theta \xleftarrow{\$} \mathscr{R}_R$; $C_{\mathrm{Rcb},\theta} := C_{\mathrm{cb},\theta} \cdot \mathsf{Enc}(pk, R_\theta)$
  $\mu := \mu - t$
  if $\mu \neq 0$ **then**
    $\theta := \theta + 1$
return $(\theta, \{C_{Rcb,\iota}\}_{\iota=1}^\theta, \{R_\iota\}_{\iota=1}^\theta)$

Theorem 1. *Suppose that the Paillier PKE scheme* Pai *is secure and the SFE scheme* SF *is secure, then the YJ scheme with a database* D *is client-secure with*

$$\mathsf{Adv}^{\mathsf{CP}}_{\mathsf{YJ},\mathscr{A}}(\kappa, \mathsf{D}) \leq (d\lambda) \cdot ((N+1) \cdot \mathsf{Adv}^{\mathsf{ind\text{-}cpa}}_{\mathsf{Pai},\mathscr{B}}(\kappa) + \frac{M}{2} \cdot \mathsf{Adv}^{\mathsf{pri.ind}}_{\mathsf{SF},\mathscr{E},\Phi}(\kappa, 1)).$$

Proof of Theorem 1. The proof is given following the game-based approach [23], which is summarized in Table 4. We use a superscript '$*$' to denote an element of the test oracle.

Game 0. The first game is the real security experiment, i.e. all queries in this game are simulated honestly in terms of the protocol specification. In particular, $\mathscr{C}$ uses the real fingerprint $F^* = \{f_\iota^*\}_{\iota=1}^N \xleftarrow{\$} \mathsf{FPTSim}(i^*)$ to generate ciphertexts $\{\{C_{j,0}^*\}_{j=1}^N, C_1^*\}$ and $\{C_{\mathrm{Rcb},\iota}^*\}_{\iota=1}^\theta$. Thus we have that $\mathsf{Adv}^{\mathsf{CP}}_{\Pi,\mathscr{A}}(\kappa, \mathsf{D}) = \mathsf{Adv}_0$.

Game 1. This game proceeds as before, but $\mathscr{C}$ aborts if it fails to guess the test oracle. Since there are $\lambda \cdot d$ clients' oracles at all, the probability of a correct guess is at least $1/(d \cdot \lambda)$. Thus we have that $\mathsf{Adv}_0 \leq (d \cdot \lambda) \cdot \mathsf{Adv}_1$.

Game	Description & Modification
0	Real experiment. $\{\{C^*_{j,0}\}_{j=1}^N, C^*_1\}$ and $\{C^*_{\mathrm{Rcb},\iota}\}_{\iota=1}^{\theta}$ of the test oracle are computed with $F^* = \{f^*_\iota\}_{\iota=1}^N \xleftarrow{\$} \mathsf{FPTSim}(i^*)$.
1	Abort if the challenger fails to guess the test oracle.
2	$\{C^*_{\iota,0}\}_{\iota=1}^N$ are computed with $F^{*\prime} = \{f^{*\prime}_\iota\}_{\iota=1}^N$, but $\{C^*_{\mathrm{Rcb},\iota}, C^*_1\}_{\iota=1}^{\theta}$ are computed with $F^* = \{f^*_\iota\}_{\iota=1}^N$, where $f^{*\prime}_1 \neq f^*_1$ but $\{f^{*\prime}_\iota\}_{\iota=2}^N = \{f^*_\iota\}_{\iota=2}^N$.
3.j $j \in [N]$	Game 2= Game 3.1 In Game 3.j: $f^{*\prime}_\iota \neq f^*_\iota$ for $1 \leq \iota \leq j$, but $\{f^{*\prime}_\iota\}_{\iota=j+1}^N = \{f^*_\iota\}_{\iota=j+1}^N$.
4	Generating C^*_1 using a random squared RSS values. $\forall\{\{C^*_{j,0}\}_{j=1}^N, C^*_1\}$ and $\{C^*_{\mathrm{Rcb},\iota}\}_{\iota=1}^{\theta}$ are independent now.
5	A random location is chosen to answer the **LocTest** query

Table 4: Sequence of games for client-privacy.

Game 2. In this game, $\mathscr{C}$ changes the encrypted location query of the test oracle based on a fingerprint $F^{*\prime}$ whose first RSS value $f^{*\prime}_1$ is distinct to $f^*_1 \in F^*$ (chosen in **InitHonestO**$(\tau^*, s^*, i^*, rds^*)$ query). And we require that $F^{*\prime}$ results in a location within an area determined by $\mathsf{ins}^{s^*}_{\tau^*}$. This requirement can be achieved by repeating the sampling procedure of $F^{*\prime}$. However, the SFE protocol instance of $\pi^{s^*}_{\tau^*}$ is still run based on the real distances $\{\overline{d^*}_\iota\}_{\iota=1}^{\theta}$ which are generated based on F^* and the fingerprint database D and the random values of S's oracle. If there exists an adversary $\mathscr{A}$ distinguishing the real location (related to the encrypted location query) from a fabricated one, then we can use it to build an efficient algorithm $\mathscr{B}$ to break the security of Pai. Note that the encrypted location query with $F^{*\prime}$ may imply a random location in the specified area.

The main reduction idea is to let $\mathscr{B}$ submit $(-2f^*_1, -2f^{*\prime}_1)$ to the PKE challenger $\mathscr{C}_{\mathsf{Pai}}$ to obtain the challenge ciphertext C^*_{Pai}, and set $C^*_{1,0} = C^*_{\mathsf{Pai}}$. The rest of the ciphertexts in the location query of the test oracle are generated as before. Next, $\mathscr{B}$ computes a distance set $\{\overline{d^*_\iota}\}_{\iota=1}^{\theta}$ based on F^*, D and $\{R^*_\iota\}_{\iota=1}^{\theta}$ of her own choice. Then, $\mathscr{B}$ generates $\{C^*_{\mathrm{Rcb},\iota}\}_{\iota=1}^{\theta}$ based on $\{\overline{d^*_\iota}\}_{\iota=1}^{\theta}$ instead. The simulation of SFE protocol is performed using $\{\overline{d^*_\iota}\}_{\iota=1}^{\theta}$ and $\{R^*_\iota\}_{\iota=1}^{\theta}$. Other oracles and the corresponding queries are simulated by $\mathscr{B}$ as the previous game using the secrets of her own choice.

If C^*_{Pai} encrypts $-2f^{*\prime}_1$ then the game is equivalent to this game since out^*_1 is a 'fabricate' location that is not computed based on the encrypted location query. Otherwise it is identical to the previous game. Applying the security of Pai, we therefore obtain that $\mathsf{Adv}_1 \leq \mathsf{Adv}_2 + \mathsf{Adv}^{\mathsf{ind\text{-}cpa}}_{\mathsf{Pai},\mathscr{B}}(\kappa)$.

Note that the ciphertexts $\{C^{*\prime}_{\mathrm{Rcb},\iota}\}_{\iota=1}^{\theta}$ (cf. $\{\overline{d^*_\iota}\}_{\iota=1}^N$) received by the test oracle are irrelevant to $C^*_{1,0}$ and C^*_1 (cf. f^*_1 in F^*). In the following, we are going to modify the game to let the ciphertexts $\{C^{*\prime}_{\mathrm{Rcb},\iota}\}_{\iota=1}^{\theta}$ are independent of all ciphertexts (i.e., C^*_1 and $\{C^*_{\iota,0}\}_{\iota=1}^N$) in the location query.

Game 3. In this game, we choose a random fingerprint $F^* \xleftarrow{\$} \mathsf{FPTSim}(i^*)$ and a random fingerprint $F^{*\prime} \xleftarrow{\$} \mathscr{R}_v^N$ such that $F^{*\prime}$ results in a location area determined by $\mathsf{ins}^{s^*}_{\tau^*}$. The ciphtertexts $\{\{C^*_{\iota,0}\}_{\iota=1}^N, C^*_1\}$ of the test oracle are computed based on $F^{*\prime}$. But the ciphertexts $\{C^*_{\mathrm{Rcb},\iota}\}_{\iota=1}^{\theta}$ are computed based on F^*, D and $\{R^*_\iota\}_{\iota=1}^{\theta}$. In order to show that no adversary can distinguish the above modification, we first define a series of sub-games (**Game** 3.1, **Game** 3.2, ..., **Game** 3.N), where **Game** 3.1 equals to **Game** 2, and **Game** 3.N equals to this game.

In **Game** 3.j (for $2 \leq j \leq N-1$)), two fingerprints $F^{*\prime}$ and F^* are generated (as before) such that the RSS value sets $\{f_\iota\}_{\iota=1}^j \in F^*$ and $\{f_\iota{}'\}_{\iota=1}^j \in F^{*\prime}$ are chosen to be pairwise distinct, and $\{f^*_\iota\}_{\iota=j+1}^N \in F^*$ and $\{f^{*\prime}_\iota\}_{\iota=j+1}^N \in F^{*\prime}$ are identical. Similarly, F^* is used to generate $\{\{C^*_{\iota,0}\}_{\iota=1}^N, C^*_1\}$ while $F^{*\prime}$ is used to generate $\{C^{*\prime}_{\mathrm{Rcb},\iota}\}_{\iota=1}^{\theta}$. Analogously, if there exists an adversary which can distinguish **Game** 3.j from **Game** 3.$(j-1)$, then it can be used to break the PKE scheme Pai.

The reduction is quite similar to the previous game. Namely, we submit $(-2f^*_j, -2f^{*\prime}_j)$ to the PKE

challenger, receiving back a challenge ciphertext C^*_{Pai} which will be assigned to $C^*_{j,0}$. If C^*_{Pai} is an encryption of $-2f^*_j$, then the simulation is identical to **Game** $3.j$, otherwise it equals to **Game** $3.(j-1)$. Since there are $N-1$ ciphertexts are changed in this game (comparing with the previous game), we have that $\mathsf{Adv}_2 \leq \mathsf{Adv}_3 + (N-1)\mathsf{Adv}^{\mathsf{ind\text{-}cpa}}_{\mathsf{Pai},\mathscr{B}}(\kappa)$.

Game 4. In this game, we change the computation of C^*_1 by encrypting a random instead. This will lead C^*_1 to be also independent of the fingerprint in the location query. With the similar proof argument in the previous game, we have that $\mathsf{Adv}_3 \leq \mathsf{Adv}_4 + \mathsf{Adv}^{\mathsf{ind\text{-}cpa}}_{\mathsf{Pai},\mathscr{B}}(\kappa)$.

Game 5. In this game, the **LocTest** query returns a random location $L^{*\prime}$ (related to the area specified by the adversary). The challenger $\mathscr{C}$ chooses a random masked distance set with an appropriate random number set $\{R^*_l\}^{\theta}_{l=1}$ to generate $\{C^*_{\mathrm{Rcb},l}\}^{\theta}_{l=1}$. However, $\mathscr{C}$ simulates the SFE protocol view of the test oracle using a simulator $\mathscr{S}(1^\kappa, L^{*\prime}, \Phi(\hat{f}))$ without $\{R^*_l\}^{\theta}_{l=1}$.

Specifically, $\mathscr{E}$ first randomly chooses a combined distance set $x = \{\overline{d^*_l}\}^N_{l=1}$, and produces a circuit $\hat{f}$. Then, $\mathscr{E}$ asks a $\mathsf{Excute}_{\mathsf{SF}}(b,1,x,\hat{f})$ query to obtain $\mathsf{vw}^*_1 = (T^*_1, \gamma^*_1)$, where T^*_1 will be appended to $T^{s^*}_{\tau^*}$. However, $\mathscr{E}$ will abort if out^*_1 implies a location which is out of the area determined by $\mathsf{ins}^{s^*}_{\tau^*}$, since $\mathscr{E}$ does not know the random values $\{R^*_l\}^{\theta}_{l=1}$ which are chosen by the $\mathsf{Excute}_{\mathsf{SF}}(b,1,x,\hat{f})$ query. We here can only expect that the randomly chosen x could result in a location $L^{*\prime}$ within the area of $\mathsf{ins}^{s^*}_{\tau^*}$. The lower bound of the abort probability is about $2/M < \#\mathsf{ins}^{s^*}_{\tau^*}/M$. If $\mathscr{E}$ does not abort, it would simulate the SFE protocol instance of the test oracle and its S's partner oracle using vw^*_1 and its own secrets. Due to the security of SF, we have that $\mathsf{Adv}_4 \leq \mathsf{Adv}_5 + \frac{M}{2} \cdot \mathsf{Adv}^{\mathsf{pri.ind}}_{\mathsf{SF},\mathscr{E},\Phi}(\kappa,1)$.

Note that in this game the location returned by the **LocTest** query is a truly random value which is independent of the bit b chosen by the **LocTest** query and any protocol messages. Thus, the advantage that the adversary wins in this game is $\mathsf{Adv}_5 = 0$.

Putting all together the probabilities from Game 0 to Game 5, and obtain the overall result of this theorem.

Theorem 2. *Suppose that the SFE scheme* SF *is secure, the database* D *is hard to forge, then the YJ scheme is server secure with*

$$\mathsf{Adv}^{\mathsf{SP}}_{\mathsf{YJ},\mathscr{A}}(\kappa,\mathsf{D},\mathsf{LT},\rho,\alpha,\phi) \leq d\cdot\ell\cdot\mathsf{Adv}^{\mathsf{pri.ind}}_{\mathsf{SF},\mathscr{E},\Phi}(\kappa,2) + \frac{\theta\cdot d\cdot\ell}{2^\kappa} + \mathsf{Adv}^{\mathsf{DBH}}_{\mathscr{D}}(\kappa,\mathsf{D},\mathsf{LT},\rho,\alpha,\phi).$$

Game	Description & Modification
0	Real experiment.
1	Abort if two random values are equal.
2	The random values used to generate the ciphertexts $\{C^*_{\mathrm{Rcb},l}\}^{\theta}_{l=1}$ and corresponding SFE protocol instance are different.
3	Apply database entropy assumption as Definition 5.

Table 5: Sequence of games for server-privacy.

Proof of Theorem 2. We summarize the proof of this theorem in Table 5. The proof proceeds via the following games.

Game 0. The first game is the real security experiment. Thus we have that $\mathsf{Adv}^{\mathsf{SP}}_{\Pi,\mathscr{A}}(\kappa,\mathsf{D}) = \mathsf{Adv}_0$.

Game 1. In this game, the challenger aborts if two random values used to blind the distance are identical. Recall that the decrypted distances $\overline{d_i}$ $(i \in [\theta])$ are masked by fresh random numbers $\overline{R_i}$ $(i \in [\theta])$ chosen by a server's oracle. We claim $\overline{d_i}$ is collision-free with overwhelming probability about $\frac{1}{2^\kappa}$. In addition, an adversary cannot obtain any useful information from the blinded distance since a modular n operation is implicitly used in the generation of a blind distance. Therefore, we have that $\mathsf{Adv}_0 \leq \mathsf{Adv}_1 + \frac{\theta\cdot d\cdot\ell}{2^\kappa}$.

Game 2. In this game, the challenger chooses valid random values (to result in a location in the specified area), which are different from the ones used in the ciphertexts, to execute the SFE instances.

Any adversary which can distinguish this game from the previous game can be used to break the party 2's security of SF. Since there are $d \cdot \ell$ instances, we have that $\mathsf{Adv}_1 \leq \mathsf{Adv}_2 + d \cdot \ell \cdot \mathsf{Adv}^{\mathsf{pri.ind}}_{\mathsf{SF},\mathscr{E},\Phi}(\kappa, 2)$.

Game 3. Due to previous modifications, the adversaries $\mathscr{A}$ (in this game) cannot gain overwhelming advantages via the protocol messages, so that it can only generate the fabricated database based on other attack powers. If the database D used by the YJ scheme satisfies the database hardcore assumption (as Definition 5), $\mathscr{A}$ cannot output a valid fabricated database as well. Therefore, we have $\mathsf{Adv}_2 \leq \mathsf{Adv}_3 = \mathsf{Adv}^{\mathsf{DBH}}_{\mathscr{D}}(\kappa, \mathsf{D}, \mathsf{LT}, \rho, \alpha, \phi)$.

Putting all together the advantages in the above games, we have the overall results of this theorem.

6 Conclusion

We presented the first formal privacy model for Wifi fingerprint based PPIL schemes, where both client- and server- privacy are formulated in a unilateral-malicious setting to cover state-of-the-art active attacks. The client-privacy is defined based on the classic notion of indistinguishability, and the server privacy is defined in a computational manner. The proposed model is verified by applying it for proving a recent PPIL protocol. An interesting open question here is whether or nor our security analysis approach can be applied to prove other kinds of privacy-preserving schemes which have a similar construction (i.e., using Paillier PKE and SFE) to the YJ scheme, e.g., the protocols for face recognition [24, 25]. For theoretical interesting, the reader is encouraged to define a stronger security model in the full malicious setting based on our model, and to proposed PPIL protocols which can be proven secure under such model. For example, one could allow the active adversaries to send her own messages to oracles (masquerading as either client or server). In the future work, it is also required to formally study the complexity of Definition 5. Nevertheless, it might be also interesting to consider whether or not it is possible to model the server-privacy based on indistinguishability.

Acknowledgments

This work was funded by the National Natural Science Foundation of China (Grant No. 61872051), the research project of the Humanities and Social Sciences of the Ministry of Education of China (Grant No. 16YJC870018), and the INSURE project (303578) of Academy of Finland.

References

[1] H. Liu, H. Darabi, P. Banerjee, and J. Liu, "Survey of wireless indoor positioning techniques and systems," *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, vol. 37, no. 6, pp. 1067–1080, November 2007.

[2] A. Ferreira, D. Fernandes, A. Catarino, and J. Monteiro, "Localization and positioning systems for emergency responders: a survey," *IEEE Communications Surveys Tutorials*, vol. 19, no. 4, pp. 2836 – 2870, February 2017.

[3] T. Roos, P. Myllymäki, H. Tirri, P. Misikangas, and J. Sievänen, "A probabilistic approach to wlan user location estimation," *International Journal of Wireless Information Networks*, vol. 9, no. 3, pp. 155–164, 2002.

[4] K. Kaemarungsi and P. Krishnamurthy, "Modeling of indoor positioning systems based on location fingerprinting," in *Proc. of the 23th IEEE International Conference on Computer Communications (INFOCOM'04), HongKong, China*. IEEE, March 2004, pp. 1012–1022.

[5] E. Elnahrawy, X. Li, and R. P. Martin, "The limits of localization using signal strength: A comparative study," in *Proc. of the First Annual IEEE Communications Society Conference on Sensor and Ad Hoc Communications and Networks (SECON'04), Santa Clara, California, USA*. IEEE, October 2004, pp. 406–414.

[6] N. Swangmuang and P. Krishnamurthy, "Location fingerprint analyses toward efficient indoor positioning," in *Proc. of 6th Annual IEEE International Conference on Pervasive Computing and Communications (PerCom'08), Hong Kong*. IEEE, March 2008, pp. 100–109.

[7] V. Honkavirta, T. Perala, S. Ali-Loytty, and R. Piché, "A comparative survey of wlan location fingerprinting methods," in *Proc. of the 6th Workshop on Positioning, Navigation and Communication (WPNC'09), Hannover, Germany*. IEEE, March 2009, pp. 243–251.

[8] A. M. Hossain and W.-S. Soh, "Cramer-Rao bound analysis of localization using signal strength difference as location fingerprint," in *Proc. of the 29th IEEE International Conference on Computer Communications (INFOCOM'10), San Diego, California, USA*. IEEE, March 2010, pp. 1–9.

[9] J. Talvitie, M. Renfors, and E. S. Lohan, "Distance-based interpolation and extrapolation methods for RSS-based localization with indoor wireless signals," *IEEE Transactions on Vehicular Technology*, vol. 64, no. 4, pp. 1340–1353, 2015.

[10] S. Li, H. Li, and L. Sun, "Privacy-preserving crowdsourced site survey in wifi fingerprint-based localization," *EURASIP Journal on Wireless Communications and Networking*, vol. 2016, no. 1, p. 123, May 2016.

[11] S. He and S. H. G. Chan, "Wi-fi fingerprint-based indoor positioning: Recent advances and comparisons," *IEEE Communications Surveys Tutorials*, vol. 18, no. 1, pp. 466–490, August 2016.

[12] Z. Yang and K. Järvinen, "The death and rebirth of privacy-preserving wifi fingerprint localization with paillier encryption," in *Proc. of the 37th IEEE International Conference on Computer Communications (INFOCOM'18), Honolulu, Hawaii, USA*. IEEE, April 2018, pp. 1223–1231.

[13] H. Li, L. Sun, H. Zhu, X. Lu, and X. Cheng, "Achieving privacy preservation in wifi fingerprint-based localization," in *Proc. of the 33th IEEE International Conference on Computer Communications (INFOCOM'14), Toronto, Canada*. IEEE, April 2014, pp. 2337–2345.

[14] A. Konstantinidis, G. Chatzimilioudis, D. Zeinalipour-Yazti, P. Mpeis, N. Pelekis, and Y. Theodoridis, "Privacy-preserving indoor localization on smartphones," *IEEE Transactions on Knowledge and Data Engineering*, vol. 27, no. 11, pp. 3042–3055, November 2015.

[15] T. Zhang, S. S. M. Chow, Z. Zhou, and M. Li, "Privacy-preserving wi-fi fingerprinting indoor localization," in *Proc. of the 11th International Workshop on Security (IWSEC'16), Tokyo, Japan*, ser. Lecture Notes in Computer Science, vol. 9836. Springer International Publishing, September 2016, pp. 215–233.

[16] Z. Yang and K. Järvinen, "The death and rebirth of privacy-preserving wifi fingerprint localization with paillier encryption," 2018, http://eprint.iacr.org/2018/259.pdf [Online; accessed on March 25, 2019].

[17] P. Paillier, "Public-key cryptosystems based on composite degree residuosity classes," in *Proc. of the 1999 International Conference on the Theory and Application of Cryptographic Techniques (EUROCRYPT'99), Prague, Czech Republic*, ser. Lecture Notes in Computer Science, vol. 1592. Springer Berlin Heidelberg, May 1999, pp. 223–238.

[18] M. Bellare and P. Rogaway, "Entity authentication and key distribution," in *Proc. of the 13th Annual International Cryptology Conference (CRYPTO'93), Santa Barbara, California, USA*, ser. Lecture Notes in Computer Science, vol. 773. Springer, Berlin, Heidelberg, July 1993, pp. 232–249.

[19] E. S. Lohan *et al*, "Indoor WLAN measurement data," 2014, http://www.cs.tut.fi/tlt/pos/MEASUREMENTS_WLAN_FOR_WEB.zip [Online; accessed on March 25, 2019].

[20] M. Bellare, V. T. Hoang, and P. Rogaway, "Foundations of garbled circuits," in *Proc of the 2012 ACM Conference on Computer and Communications Security (CCS'12), New York, New York, USA*. ACM, October 2012, pp. 784–796.

[21] M. Bellare, V. T. Hoang, and S. Keelveedhi, "Efficient garbling from a fixed-key blockcipher," in *Proc. of the 34th IEEE Symposium on Security & Privacy (S&P'13), San Francisco, California, USA*. IEEE, May 2013, pp. 478–492.

[22] T. Jager, F. Kohlar, S. Schäge, and J. Schwenk, "On the security of TLS-DHE in the standard model," in *Proc. of the 32nd Annual Cryptology Conference (Crypto'12), Santa Barbara, California, USA*, ser. Lecture Notes in Computer Science, vol. 7417. Springer Berlin Heidelberg, Auguest 2012, pp. 273–293.

[23] V. Shoup, "Sequences of games: a tool for taming complexity in security proofs," Cryptology ePrint Archive, Report 2004/332, 2004, http://eprint.iacr.org/ [Online; accessed on March 25, 2019].

[24] A.-R. Sadeghi, T. Schneider, and I. Wehrenberg, "Efficient privacy-preserving face recognition," in *Proc. of the 12th International Conference on Information, Security and Cryptology (ICISC'09), Seoul, Korea*, ser. Lecture Notes in Computer Science, vol. 5984. Springer Berlin Heidelberg, December 2010, pp. 229–244.

[25] M. Blanton and P. Gasti, "Secure and efficient protocols for iris and fingerprint identification," in *Proc. of the 16th European Symposium on Research in Computer Security (ESORICS'11), Leuven, Belgium*, ser. Lecture Notes in Computer Science, vol. 6879. Springer Berlin Heidelberg, September 2011, pp. 190–209.

Author Biography

Zheng Yang received the Master degree from Chongqing University, in 2009. He received his Ph.D. degree from Horst Görtz Institute for IT Security, Ruhr-University Bochum, in 2013. He is currently a Post-doc researcher with the Singapore University of Technology and Design. His main research interests include information security and cryptography.

Kimmo Järvinen received the M.Sc. (Tech.) and D.Sc. (Tech.) degrees in electrical engineering from the Helsinki University of Technology (TKK) in Finland in 2003 and 2008, respectively. From 2008 to 2013 and from 2015 to 2016, he was a postdoctoral researcher in the Department of (Information and) Computer Science in Aalto University in Finland. From 2014 to 2015, he was with the COSIC Group in KU Leuven ESAT in Belgium. Since 2016, he has been a Senior Researcher with the Department of Computer Science in University of Helsinki in Finland. His research interests lie in the domains of security and cryptography and, especially, in developing efficient and secure implementations of cryptosystems.