A Cryptographic Protocol for Efficient Mutual Location Privacy Through Outsourcing in Indoor Wi-Fi Localization

Samuel N. Eshun and Paolo Palmieri, *Member, IEEE*

***Abstract*— Digital services and applications are increasingly requiring location information from users to provide personalized services. However, disclosing one's location introduces significant privacy risks, as location traces are highly unique and can be used to infer additional sensitive data. While location-based services were once restricted to outdoor spaces, given the lack of GPS signal indoors, a growing number of applications rely on Wi-Fi to provide indoor localization. Indoor localization can impact privacy to an even greater degree, as most of our daily activities occur indoors. Therefore, several indoor privacy protocols have been proposed, focusing on protecting the user's location. However, the problem of mutual location privacy, that is, the protection of both the user's privacy and the service provider's location database, has not been addressed, particularly against malicious (active) adversaries. In addressing this gap, this paper presents an efficient and privacy-preserving cryptographic protocol for indoor localization. Our protocol hides the user's location, while also protecting the service provider's location map and areas of interest against malicious users. Furthermore, the protocol outsources most of the user-side heavy computation to a third-party cloud server, which does not need to be trusted by the parties as it remains oblivious to both user's location and the provider's database throughout the computations. Compared to leading solutions in the literature, including Eshun and Palmieri (2019) and Li et al. (2014), our protocol is the first to provide security against malicious users. Additionally, it significantly reduces the user computation and communication overhead (of up to 99%), making it potentially the first practicable scheme in resource-constrained mobile and IoT environments.**

***Index Terms*— Location privacy, garbled circuit, oblivious transfer, homomorphic encryption, bloom filter.**

I. Introduction

A GROWING trend in digital services is personalization, where services are tailored based on user context, such as location [3]. Location-Based Services (LBS) utilize user devices with Global Navigation Satellites System (GNSS) receivers, such as GPS and Galileo, to determine user location. However, GNSS signals are weak indoors [4], prompting the development of indoor localization techniques, with Wi-Fi emerging as a prominent solution due to existing infrastructure. Indoor Wi-Fi localization involves two phases: offline and online. In the offline phase, a service provider (SP) constructs a *radio-map* database using Wi-Fi *received signal strength* (RSS) measurements at known reference points. In the online phase, a user's RSS measurements are compared with the radio-map to compute their location using various distance metrics [5].

This personalization introduces privacy challenges, as it requires disclosing user information. Preserving location privacy in indoor LBS is particularly challenging due to the necessity of SP involvement, which implies potential disclosure of sensitive user and SP data. In this paper, we focus on *mutual location privacy*, aiming to protect both user location and SP's sensitive data, aligning with the concept of secure multi-party computation (MPC).

MPC, a cryptographic technique preserving privacy during joint computations, can be applied in indoor Wi-Fi localization protocols. However, MPC's computational demands may limit its scalability, especially for resource-constrained users. Outsourcing user computation to cloud servers can alleviate this bottleneck, offering high computational power and storage, thereby enhancing the efficiency and privacy of indoor localization protocols [6], [7], [8].

A critical, underexplored question is how an SP can securely verify a user's presence in predefined areas without learning their location outside these areas. Examples include employee attendance verification. In a healthcare context, hospitals may seek to gather data on the average duration a patient spends in a specific ward for a particular service. This data could also be valuable to health insurance companies aiming to compare hospitals and adjust their reimbursement strategies based on the value of care provided [9]. To uphold patient privacy, monitoring should be confined to designated areas (e.g., a scan room), ensuring that the provider remains unaware of the patient's location outside these zones. Simultaneously, it is imperative to secure the building's map and specific indoor locations to safeguard sensitive infrastructure information.

In emergencies, such as evacuations, or for contact tracing during pandemics like COVID-19, it may be necessary to track individuals within specific areas while preserving their privacy outside these zones. This paper addresses these challenges, proposing a solution that balances effective localization with robust privacy protection.

Manuscript received 22 August 2023; revised 12 January 2024; accepted 15 February 2024. Date of publication 1 March 2024; date of current version 2 May 2024. This work was supported in part by the Science Foundation Ireland under Grant 13/RC/2077_P2 (CONNECT Phase 2). The associate editor coordinating the review of this manuscript and approving it for publication was Dr. Alptekin Küpçü. *(Corresponding author: Samuel N. Eshun.)*

The authors are with the School of Computer Science and IT, University College Cork, Cork, T12 XF62 Ireland (e-mail: s.eshun@cs.ucc.ie; p.palmieri@cs.ucc.ie).

Digital Object Identifier 10.1109/TIFS.2024.3372805

A. Related Work

In this paper, we categorize relevant works in location privacy preservation into two primary approaches: homomorphic encryption and dual secure computations. We apply the concept of outsourced secure multi-party computation to location privacy, a concept previously validated in various settings, such as in Araki et al. [10].

Li et al. [2] proposed a construction based on homomorphic encryption for privacy-preserving indoor localization. However, their security model has vulnerabilities, as indicated by results in [11], which show potential exposure of the service provider's database. Zhang et al. [12] also employ homomorphic encryption, but their protocol is not secure against a malicious user.

The work in [1] introduces a unique perspective, allowing service providers to privately determine if a user is within predefined areas, while the user can learn their exact location privately. The second approach, exemplified by Shu et al. [13] and Yang et al. [14], combines two secure computations, such as Paillier encryption and oblivious transfer, to preserve location privacy. These protocols, however, face challenges in computational efficiency and security against malicious entities.

Recent works, like Järvinen et al. [6] and Oleynikov et al. [7], leverage Secure Multi-Party Computation (SMPC) to offload computations to non-colluding, semi-honest parties, enhancing protocol efficiency while maintaining security.

This paper aims to balance the privacy of both the user and the service provider (SP). Specifically, our protocol enables the SP to securely verify a user's presence in predefined areas while preserving the user's privacy elsewhere. This is achieved through the involvement of non-colluding servers, a widely accepted assumption in the literature [6], [15], [16], [17]. We argue that the non-collusion assumption is practical and motivated by legal, contractual, and reputational incentives for both the SP and the cloud server to uphold user data privacy and security. While collusion between the service provider and the cloud server is conceivable, the risks and legal obligations associated with user data privacy and security serve as potent deterrents. These factors strongly motivate both parties to adhere to the protocol and avoid collusion. For instance, in scenarios where a business outsources data processing to a third-party cloud service provider, the cloud server may execute computations, but data ownership remains with the company, further reducing the likelihood of collusion.

B. Contribution

This paper presents an efficient and secure indoor location privacy protocol that allows the service provider to query if a user is within an area of interest (out of a predefined set of areas) while maintaining the exact location of the user's privacy. As such, the provider does not learn the location of the user but only the identifier of the area of interest if the user is within one and nothing otherwise. The scheme prevents providers from defining too many or too broad areas, while the provider's database is also protected from malicious users.

The proposed protocol supersedes previous works in the indoor Wi-Fi location privacy domain, including [1], where the protocol offers protection to both the user's fingerprint, and the service provider's database yet delivers the service based on the users' physical location. Unlike [1], where the security model was limited by the assumption of honest-but-curious users, this protocol's security is assured even against a malicious user. Our work combines state-of-the-art techniques in secure multi-party computation to preserve the privacy of the SP database against malicious users. The model builds upon the spatial Bloom filter data structure [18], [19], [20] and substantially improves the overhead at the user-side with respect to comparable protocols in the literature. Specifically, the main contributions of this paper are summarized below:

- We design a hybrid technique by combining Damgård, Geisler and Krøigaard (DGK) [21], [22] encryption and an optimized garbled circuit to preserve both parties' privacy. DGK encryption protects the user's query data and significantly reduces the message size space compared to other well-known additive homomorphic encryption schemes like Paillier (used, for instance, in [10]). The garbled circuit [23], [24], on the other hand, preserves the privacy of the database against malicious users.
- Our protocol provides protection against malicious users with the addition of a single server (CS). This method is a significant improvement over comparable protocols, such as [6], which requires two parties to achieve comparable security levels.
- Our protocol significantly improves the computation overhead for encrypting the fingerprint compared to solutions in the literature (as discussed in Section V) and offloads almost all the computation overhead at the retrieval phase to the cloud server. In addition, the communication overhead is reduced by over 99.99% at the user-side compared to the non-outsourced protocols. We achieved this by delegating most of the heavy computation of the garbled circuit at the user-side to the cloud server, thus making the protocol suitable for practical applications when compared to computationally intensive user/server proposals in [1], [2], and [11].
- We provide a security and complexity analysis of our protocol and quantify the computation and bandwidth gains compared to other similar localization protocols available in the literature (Sections IV and V).

II. Preliminaries

This section introduces the primitives employed in the design of the proposed protocol (presented in Section III). Table I provides the notation and symbols used in this paper.

A. Homomorphic Encryption (HE)

Homomorphic encryption schemes, which are semantically secure (IND-CPA), enable specific computations on encrypted data. This paper's protocol leverages the additive property of such encryption, allowing addition between encrypted data and multiplication by a plaintext constant. Given plaintexts a, c, and d, with encryption and decryption functions $\mathsf{Enc}()$,

TABLE I
NOTATION: ACRONYMS AND SYMBOLS

Symbol	Meaning
RSS	Received signal strength
$\mathbf{n}$	The number of access points (ap)
$\mathbf{m}$	The number of reference points
$r_{i,j}$	RSS value at position i for ap j
V_i	RSS values for all n ap's at i
V_u	RSS values of user (u)
c	Encrypted RSS values of user
SBF	The spatial Bloom filter
∇	An area of interest (AoI)
$\mathcal{T}_2$	List of reference points
ξ	Set of all the areas of interest
$B^{\#}$	SBF over the set of areas of interest
$\mathsf{Enc}(B^{\#})$	Encryption of $B^{\#}$ using pk_s
pk_u	Public key of user U
sk_u	Secret key of user U
pk_s	Public key of service provider SP
sk_s	Secret key of service provider SP
$\mathbf{T}$	The bit-length of RSA moduli
$\mathsf{Enc}(\delta)$	Unmasked encrypted euclidean distance
$\mathsf{Enc}[\delta]$	Masked encrypted euclidean distance
$\mathbb{D}$	Referenced database of service provider
π	Garbled indexes of smallest KNN distances
γ	Random noise/value
d	User exact location coordinate
B_u	Bloom filter build over ∇
$B_u^{\#}$	Output of $\mathsf{Enc}(B^{\#})$ and B_u

$\mathsf{Dec}()$, and operations $\boxplus$ and $\boxdot$ for homomorphic addition and plaintext multiplication, respectively, we have:

$$\mathsf{Enc}\,(c+d) = (\mathsf{Enc}\,(c) \boxplus \mathsf{Enc}\,(d)) \quad , \tag{1a}$$

$$\mathsf{Enc}\,(a \cdot c) = (a \boxdot \mathsf{Enc}\,(c)) \tag{1b}$$

where the operations $(+, \cdot)$ are computed in the plaintext space, and $(\boxplus, \boxdot)$ in the ciphertext space.

Paillier encryption [25] is a widely used additive homomorphic encryption scheme, and its security relies on the decisional composite residuosity problem. It has a public key size of T-bits RSA modulus $N = p \cdot q$, where p and q are two large primes. To encrypt a message x, let $\mathbb{Z}_N$ be a set of integers modulus N and $\mathbb{Z}_N^* \subseteq \mathbb{Z}_N$. Choose $g \in \mathbb{Z}_{N^2}^*$, the generator of the multiplicative group and a random number $r \in \mathbb{Z}_N^*$, then for $x \in \mathbb{Z}_N$ gives:

$$\mathsf{Enc}(x) = g^x r^N \mod N^2 \quad , \tag{2}$$

and the public key being (g, N). The reader is referred to [25] for a detailed description and the security proof of the scheme.

In order to increase efficiency with respect to Paillier-based schemes, the protocol proposed in this paper makes use of the DGK [21], [22] encryption scheme, which has a smaller ciphertext compared to Paillier due to the lower exponent of the modulus. Without any modification, DGK works only on small message space (μ) because it was initially designed for secure comparisons of integers. The DGK hardness is based on the strong assumption of the RSA subgroup, and it performs better than the other additive homomorphic encryption schemes for both encryption and decryption. Compared to Paillier, the ciphertext space is T-bits of the RSA modulus $N = p \cdot q$ while Paillier is $2T$. Similar to other cryptographic algorithms, the scheme has three main parts:

Key Generation The process of generating a key starts by first choosing RSA modulus $N = p \cdot q$, where p and q are two large primes and $\log_2 p \approx \log_2 q$. Select a plaintext space μ of size $l - bits$ and another two distinct primes of size $t - bits$; p_t and q_t such that $p_t | p - 1$ and $q_t | q - 1$. Choose $g, h \in \mathbb{Z}_N^*$ the generators of $\mathbb{Z}_N^*$, such that g and h have orders $\mu p_t q_t$ and $p_t q_t$ respectively. Therefore, the public key is (N, g, h, μ) and the secret key (p, q, p_t, q_t).

Encryption To encrypt a message $x \in \mathbb{Z}_\mu$, choose $r \in \mathbb{Z}_N$ and compute:

$$\mathsf{Enc}(x) = g^x h^r \mod N \tag{3}$$

Decryption The decryption is simply raising the ciphertext $\mathsf{Enc}(x)$ to a power $p_t q_t$ modulus N, i.e.,

$$x = \mathsf{Enc}(x)^{p_t q_t} \mod N \tag{4}$$

Due to the small message space μ, a table of $g^{p_t q_t}$ values can be built to optimize the decryption process.

Though DGK works on small message space μ, it can be modified to work on large plaintext space using the Pohlig-Helman algorithm [26] to perform the decryption.

B. Oblivious Transfer (OT)

Oblivious Transfer (OT) is a two-party protocol where one party (the *sender*) holds a secret input of n bits or strings of bits, and the other party (the *receiver*) wants to obtain one of these bits/strings. The receiver wants the sender to remain oblivious to which bit she chooses, while the sender wants to ensure only one bit is revealed to the receiver. When $n = 2$, we call it *1-out-of-2* OT. The protocol was first introduced in 1981 by Rabin [27] and has since become a fundamental component in several cryptographic constructions [28], [29], [30]. In detail, for a 1-out-of-2 OT protocol, a sender inputs m-pairs of bits $\{\mathcal{K}_i^0, \mathcal{K}_i^1\}$ and the receiver inputs her choice in the form of m-bits $b_i \in \{0, 1\}$. At the end of the protocol, the receiver obtains $\mathcal{K}_i^{b_i}$ and learns no information about $\mathcal{K}_i^{1-b_i}$. On the other hand, the sender obtains nothing and learns nothing about the receiver's choice b_i.

C. Garbled Circuit (GC)

Garbled circuits [23], [24], a work of Yao in the late 1980s, allow two parties to compute a function on their private input.

For a given Boolean circuit C, party S (the *generator*) constructs a garbled circuit G. For each wire w_i of C, S assigns two random keys $(\tilde{k}_i^0, \tilde{k}_i^1)$. For each gate G_i, S uses these keys to encrypt and create a table T_i, where each entry holds the encrypted output value. S then permutes T_i and sends it, along with the garbled circuit and the generator's input garble values, to the other party (the *evaluator*).

To evaluate the garbled circuits, the evaluator (E) and S engage in 1-out-of-2 OT to obtain E's plain inputs' garbled values from S. With these values for both parties, E evaluates the garbled circuit gate by gate, retrieving the output from the output wires. Due to table permutation, these wires reveal nothing about the internal circuitry, so E learns only the output values. A translation table converts these to plain output values

for the output party. For Yao's protocol security proof in the semi-honest model, see [31].

To enhance the original protocol's security, various optimization techniques were introduced, including free XOR [32], point-and-permute [33], [34], and garbled row reduction [35]. [36] further reduced AND gates to two ciphertexts, albeit incompatible with free XOR. This was refined by [37] using a half-gate technique, achieving two ciphertexts for AND gates and zero for XOR gates (free XOR). These advancements collectively enhance the protocol's efficiency, notably reducing communication and computation overhead in circuit construction and evaluation.

D. Bloom Filter (BF)

A Bloom Filter (BF) [38] is a space-efficient probabilistic data structure that supports set-membership queries. It represents a set $S = \{x_1, \cdots, x_n\}$ of n-elements using an array of size m-bits. Independent cryptographic hash functions $\mathcal{H} = \{h_1, \cdots, h_k\}$ are used to insert elements into the filter. Initially, the filter is set to zero. To insert an element x, each hash function $h \in \mathcal{H}$ maps x to a position $a \in \{1, \cdots, m\}$ in the filter B, and sets $B\{a\} = 1$. To query if an element y belongs to S, all hash functions in $\mathcal{H}$ must map y to positions already set to 1 in B minus the probability of a false positive. If this is true, y is likely in S. If any hash function maps y to a 0 in B, then y definitely does not belong to S, as BF does not have false negatives. The false positive probability is a function of the filter's parameters and can be calculated accordingly.

E. Spatial Bloom Filter (SBF)

Spatial Bloom Filters (SBF) [18] extend the original Bloom Filter (BF) [38] for space-efficient set membership queries. Unlike BF, SBF uses a byte array, allowing multiple sets in one filter, initially designed for spatial representations. SBF introduces *inter-set errors*, a unique false positive where an element is misattributed to a different set within the filter, in addition to the standard BF false positives. False negatives are disallowed in both. In SBF, sets are represented by integer values, replacing the classical BF's binary indicators.

Unlike the traditional BF [38], SBF is able to handle multiple sets (like complex geographical data) with priority-based encoding system while providing efficient secure data handling. For a detailed analysis of SBF and its probabilistic properties, see [19], and [20].

In an SBF, each set is represented by an integer numerical value, and such values are written into the filter during construction instead of the 1s that would be written in a classical BF. We refer to [19], and [20] for a detailed analysis of the filter and its probabilistic properties. Figure 1 shows an example of a spatial Bloom filter $\mathsf{Enc}(B_u^{\#})$; which is a product of Bloom filter (B_u) and encrypted spatial Bloom filter $\mathsf{Enc}(B^{\#})$. The encrypted filter simply means the values encoded in the filter are not in the plaintext as shown in the diagram.

III. Private Indoor Wi-Fi Localization

In this section, we present an innovative protocol for private indoor Wi-Fi-based localization. We assume familiarity

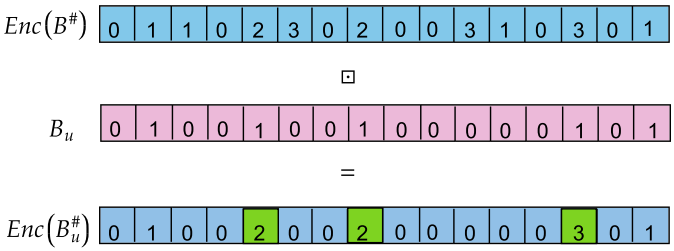


Fig. 1. Example of Spatial Bloom filter: The private Hadamard product of $\mathsf{Enc}(B^{\#})$ and B_u, producing sbf $\mathsf{Enc}(B_u^{\#})$. For clarity and ease of reading, the image depicts plaintext values (rather than the corresponding ciphertexts) for $\mathsf{Enc}(B^{\#})$ and $\mathsf{Enc}(B_u^{\#})$ (in light blue).

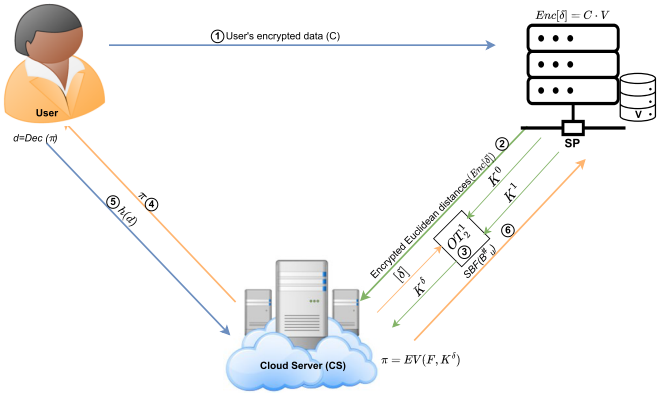


Fig. 2. System architecture for secure three-party computation. Most of the heavy user-side computation is delegated to the cloud server (CS) without revealing the user's private input. Using the OT protocol, the CS can securely interact with the service provider (SP) in order to perform secure computations on the user's behalf. The arrow's colour indicates the flow of information from one party to another, with light blue, light green, and light orange representing the user, SP, and CS, respectively.

with key primitives: homomorphic encryption (specifically, the DGK cryptosystem), Oblivious Transfer (OT), Garbled Circuits (GC), Bloom Filters (BF), and Spatial Bloom Filters (SBF) introduced in the previous section, II.

The protocol engages three parties: a user, a location service provider, and a third-party computation outsourcer. Protocol security is addressed in Section IV, and efficiency comparisons are in Section V.

Wi-Fi fingerprinting, central to this protocol, involves two phases: offline set-up and online phase. In set-up, the service provider constructs a reference database $\mathbb{D}$, creating a fingerprint (V) at each reference point, representing the received signal strength (RSS) at varying access points. For a reference point (x_i, y_i), an n-dimensional vector $V_i = \{r_1, r_2, \cdots, r_n\}$ is formed, where n is the total number of access points and m is the total number of reference points.

The fingerprints of all reference points form the reference database (radio map), used in the online phase to localize a user. Table II shows an example of this database. Symbol $\perp$ indicates a reference point outside the range of an access point (no signal received).

In the online phase, a user u seeking their location generates a fingerprint by measuring the RSS values $V_u = \{r_1, r_2, \cdots, r_n\}$ at location (x_u, y_u). This fingerprint is

TABLE II
WI-FI-FINGERPRINT REFERENCE DATABASE ($\mathbb{D}$): EACH COLUMN RELATES TO AN ACCESS POINT ap_i, WHILE EACH ROW REPRESENTS THE FINGERPRINT V_i FOR REFERENCE POINT COORDINATES (x_i, y_i). $\perp$ INDICATES NO SIGNAL FROM THE ap AT THAT LOCATION. $\mathbb{D}$ IS ALSO CALLED A RADIOMAP

	$\mathbf{ap_1}$	$\mathbf{ap_2}$	$\mathbf{ap_3}$	$\mathbf{ap_4}$	$\cdots$	$\mathbf{ap_n}$
$\mathbf{V_1}$	$r_{1,1}$	$r_{1,2}$	$\perp$	$\perp$	$\cdots$	$r_{1,n}$
$\mathbf{V_2}$	$\perp$	$r_{2,2}$	$r_{2,3}$	$r_{2,4}$	$\cdots$	$\perp$
$\mathbf{V_3}$	$\perp$	$\perp$	$r_{3,3}$	$r_{3,4}$	$\cdots$	$r_{3,n}$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
$\mathbf{V_m}$	$r_{m,1}$	$r_{m,2}$	$r_{m,3}$	$\perp$	$\cdots$	$\perp$

compared to the reference database $\mathbb{D}$ to estimate the user's position using a distance computing algorithm.

The Euclidean distance, given in Equation 5, is a commonly used algorithm. It quantifies the difference between V and V_u.

$$\delta = \|V - V_u\| = \sqrt{\sum_{i=1}^{n}(V_i - V_{u_i})^2} \tag{5}$$

A computation involving the square roots cannot be directly rendered through homomorphic computation; hence the squared Euclidean distance (equation 6), which achieves the same result, is used here instead.

$$\begin{aligned}\delta = \|V - V_u\|^2 &= \sum_{i=1}^{n}(r_i - r_{u_i})^2 \\ &= \sum_{i=1}^{n} r_i^2 + \sum_{i=1}^{n} r_{u_i}^2 + \sum_{i=1}^{n} - 2r_{u_i} \cdot r_i \cdot \end{aligned} \tag{6}$$

From equation 6, the smallest k nearest distances $\{\delta_j\}_{j=1}^{k}$ are selected and the average (k-means) is computed to localize the user.

A. Protocol Participants

Three parties participate in the proposed protocol are:

User: A mobile device that inputs RSS fingerprint V_u to a service provider to obtain its exact location.

Service Provider (SP): Wants to verify if a user is within a specified area ∇ without learning the user's location outside this area. When the user is in ∇, the SP only learns this fact, not the exact location. The SP can define multiple areas of interest $\xi = \{\nabla_1, \ldots, \nabla_l\}$.

Cloud Server (CS): A third party assisting the user with computationally intensive operations, without contributing to the evaluation function.

B. Protocol Overview

We provide a high-level overview of the protocol, detailed further later in the text, outlining each participant's operations.

The protocol initiates with the SP generating a key pair (pk_s, sk_s) and constructing a Spatial Bloom Filter (SBF) $B^{\#}$ over the set of areas of interest $\xi = \{\nabla_1, \ldots, \nabla_l\}$, as in [19]. The SP encrypts $B^{\#}$ with pk_s and sends $\mathsf{Enc}(B^{\#})$ to the Cloud Server (CS). This step can be offline.

The user creates a key pair (pk_u, sk_u), constructs their location fingerprint V_u using RSS data, and encrypts it: $c = \mathsf{Enc}(pk_u, V_u)$. The user sends c to the SP and sk_u to the CS, assuming non-collusion between CS and SP.

The SP computes a masked encrypted Euclidean distance $\mathsf{Enc}[\delta]$ using the user public key pk_u, the received fingerprint, and the reference database $\mathbb{D}$.

The SP computes function $\mathcal{F}$ (garbled circuit function) and engages CS in secure two-party computation and oblivious transfer to: 1) remove the SP's masking from $\mathsf{Enc}(\delta)$, and 2) compute the k-smallest Euclidean distances.

The CS sends the indexes of the k-nearest distances (appearing as random noise to the CS) to the user, who then localizes their exact location using these distances.

Periodically, the user computes their geocode, encodes it using a keyed hash function $\mathcal{H}^{\star}$, supplied by the SP, and sends the hash digests to the CS. The CS constructs a user filter (B_u) and computes the entry-wise homomorphic product with the SP's $(B^{\#})$. The result, sent to the SP, confirms the user's presence in an area of interest, without revealing specifics to either party. The CS remains unaware.

The protocol comprises an offline preparation phase and four online phases: measuring, computation, retrieval, and server localization. The complete protocol is depicted in Protocol 1. Algorithms 1-2 detail the perspective of each party, cross-referenced in Protocol 1 for clarity.

1) Preparation and Initialization Phase: In this phase, the SP publishes the localization-aided data $\{\mathcal{T}_1, \mathcal{T}_2\}$ of the reference database ($\mathbb{D}$) it holds to all users, where $\mathcal{T}_1 = \{ap_1, \cdots, ap_n\}$ is the access points and $\mathcal{T}_2 = \{\tau_1, \cdots, \tau_m\}$ represents the list of reference points. Importantly, it does not release the entire radio map. The user generates an encryption key pair $< pk_u, sk_u >$ using the DGK cryptosystem and publishes the public key pk_u while the server uses an encryption scheme that allows private Hadamard product to generate key pairs $< pk_s, sk_s >$. Through a secure communication channel, the user sends the decryption key sk_u (this can always be revoked if compromised) to the CS. Using a keyed hash function $(\mathcal{H}^{\star})$, the SP constructs a spatial Bloom filter $(B^{\#})$ over the areas of interest (∇) using the algorithm described in [19] where $\nabla \in \xi$, and encrypts it with the public key (pk_s) to obtain $\mathsf{Enc}(B^{\#})$. It is worth noting while ξ constitutes the set of all the areas of interest (∇), $\xi \subseteq \mathcal{T}_2$, thus the exact location of the user, $d \in \mathcal{T}_2$ but may be $\notin \xi$ depending on the position of the user. The service provider SP sends its public key (pk_s) together with the encrypted filter $\mathsf{Enc}(B^{\#})$ to the CS, and $\mathcal{H}^{\star}$ and garbled output function $\left[(\mathcal{H}(\pi^0), 0), (\mathcal{H}(\pi^1), 1)\right]$ to the user. The above process can be executed once and offline (that is, separately from the execution of the remainder of the protocol).

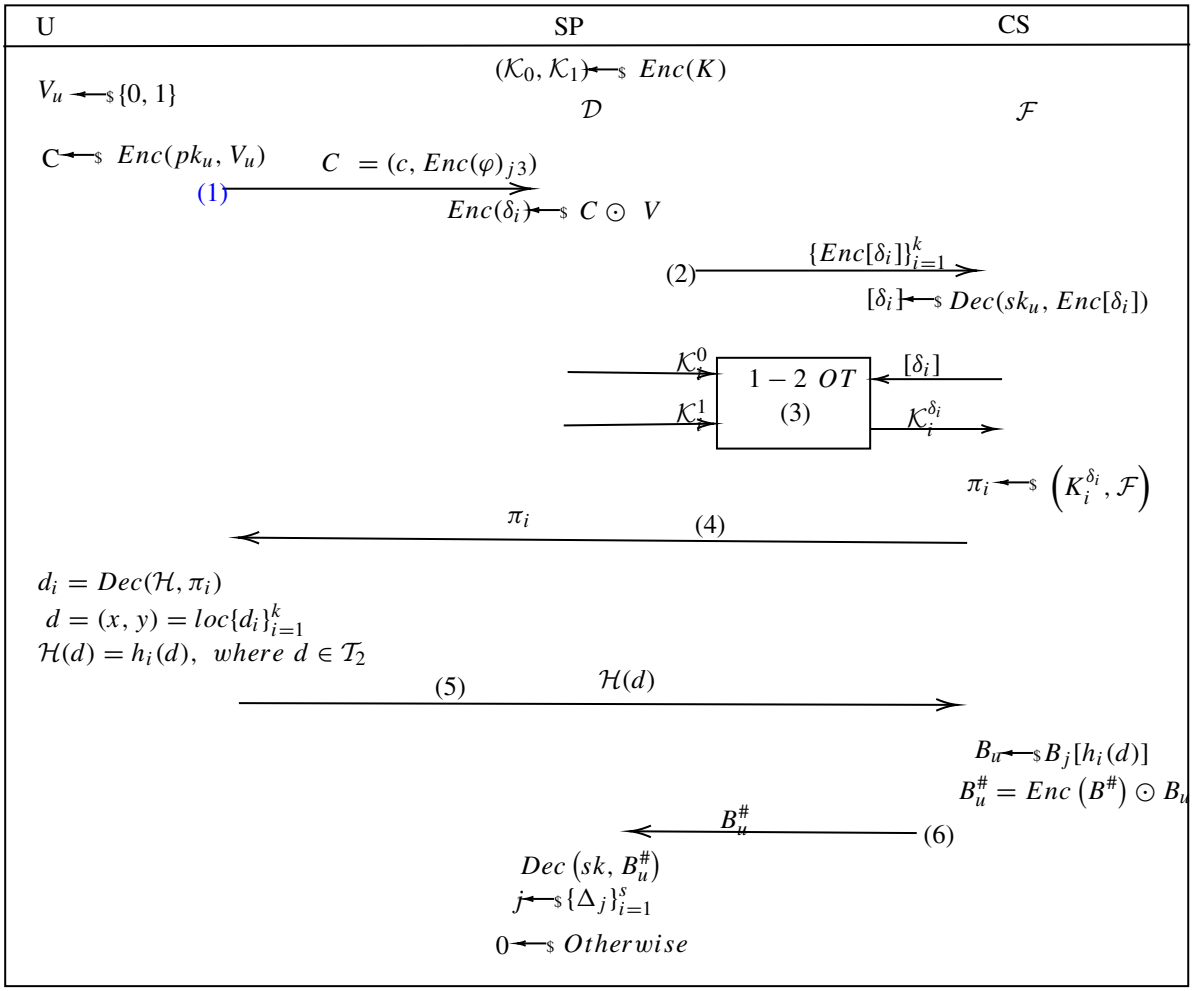
2) Measuring Phase: The user generates the RSS fingerprint $V_u = \{r_1, \cdots, r_n\}$ of her position, and computes:

$$c = \mathsf{Enc}(-2r_{u_1}), \cdots, \mathsf{Enc}(-2r_{u_n}) \tag{7a}$$

$$\mathsf{Enc}(\varphi)_{j_3} = \mathsf{Enc}\left(\sum_{i=1}^{n} r_{u_i}^2\right) \tag{7b}$$

The results are sent to the SP, as indicated in Algorithm 1 (step 1). The RSS measurement can always be done in an offline phase, and new RSS data is generated only when the user's position changes. Also, this is the only computation

Protocol 1 Outsourcing Privacy Preserving Protocol for Indoor Wi-Fi Localization



the user does before receiving the indexes of the k-smallest Euclidean distances from the CS. Interaction between the mobile user and the SP ends after the input query; the rest of the communication is either between the SP and the CS or the CS and the mobile user.

3) Computation Phase: After receiving the user's encrypted fingerprint $\{c_i\}_{i=1}^{n}$, the SP computes the masked encrypted Euclidean distances $\mathsf{Enc}[\delta]$ between the received fingerprint and the reference database $\mathbb{D}$ (Table II). The results are forwarded to the CS. The computation is possible due to the homomorphic properties of the DGK cryptosystem [21] as depicted in step (2) of Algorithm 2.

Specifically, the SP computes:

$$\mathsf{Enc}(\varphi)_{j_1} = \mathsf{Enc}\left(\sum_{i=1}^{n} r_i^2\right) \tag{8a}$$

$$\mathsf{Enc}(\varphi)_{j_2} = \prod_{i=1}^{n} \left(c_i\right)^{r_i}, \tag{8b}$$

and the encrypted euclidean distance $\mathsf{Enc}(\delta)$ such that:

$$\mathsf{Enc}(\delta) = \mathsf{Enc}\left(\sum_{i=1}^{n} (r_i - r_{u_i})^2\right) \tag{9}$$

$$= \mathsf{Enc}\left(\sum_{i=1}^{n} r_i^2\right) \cdot \prod_{i=1}^{n} (c_i)^{r_i} \cdot \mathsf{Enc}\left(\sum_{i=1}^{n} r_{u_i}^2\right) \tag{10}$$

$$= \mathsf{Enc}(\varphi)_{j_1} \cdot \mathsf{Enc}(\varphi)_{j_2} \cdot \mathsf{Enc}(\varphi)_{j_3} \tag{11}$$

where the user is responsible for computing $\mathsf{Enc}(\varphi)_{j_3}$ as shown in equation 7b, and SP computes both $\mathsf{Enc}(\varphi)_{j_1}$ and $\mathsf{Enc}(\varphi)_{j_2}$ as delineated in 8a and 8b respectively. Thus, the user computed $\mathsf{Enc}(\varphi)_{j_3}$, in conjunction with the SP computed $\mathsf{Enc}(\varphi)_{j_1}$ and $\mathsf{Enc}(\varphi)_{j_2}$, form the basis of the encrypted Euclidean distances, $\mathsf{Enc}(\delta)$.

In order to protect the database $\mathbb{D}$ from being leaked to the cloud server or a malicious user (i.e., if the user colludes with the cloud server), the service provider blinds (masks) each of the encrypted distances $\mathsf{Enc}(\delta_j)$ with a unique random value $\gamma_j \in \mathcal{R}$ such that:

$$\mathsf{Enc}\left[\delta_j\right] = \mathsf{Enc}(\delta_j) \cdot \mathsf{Enc}(\gamma_j) \tag{12}$$

where $\gamma_j \in \{0, 1\}^n \ \ \forall\ j \ \in \{1, \cdots, m\}$. The SP finally sends the masked encrypted distances $\mathsf{Enc}\,[\delta]$ to the cloud server (CS) (line 21-25 of Algorithm 2).

After receiving the masked encrypted distances, $\{\mathsf{Enc}\,[\delta]_j\}_{j=1}^{m}$, the CS decrypts these distances using the user's decryption key sk_u to obtain unencrypted masked euclidean distances $\{[\delta]_j\}_{j=1}^{m}$ made up of the euclidean distances (δ) and the added noise (γ) as shown in equation 13. The process

Algorithm 1 User (U)

Input : $V_u = \{r_1 \cdots r_n\}; r_i \in \{0,1\}^n,\ \mathcal{H}$ **Output** : B_u

USER OUTSOURCED LOCATION

1 : $C \leftarrow_\$ \emptyset$
2 : **for** $i \leftarrow_\$ 1 \cdots n$ **do**
3 : $c_i \leftarrow_\$ \mathsf{Enc}(-2r_i)$
4 : $\mathsf{Enc}(\varphi)_{j_3} \leftarrow_\$ \mathsf{Enc}\left(\sum_{i=1}^{n} r_i^2\right)$
5 : $C.append\{c, \mathsf{Enc}(\varphi)_{j_3}\}$
6 : **post** C *to* **S**

Step (1) *of Protocol* 1

RETRIEVING PRIVATE LOCATION

7 : $D \leftarrow_\$ \emptyset$
8 : **get** $\{\pi_i\}_{i=1}^{k}$ *from* **CS**
9 : $\{h_1, \cdots, h_k\} \leftarrow_\$ \mathcal{H}$
10 : **for** $j \leftarrow_\$ 1 \cdots k$ **do**
11 : $d_j \leftarrow_\$ lsb(\pi_j) \oplus h_j$
12 : **append**$\{d_j\}_j^k$ *to* D
13 : $d = (x, y) \leftarrow_\$ loc\{D\}$

Step (4) *of Protocol* 1

Algorithm 2 Service Provider (SP)

Input : $\{V_i\}_{i=1}^{n},\ \gamma \xleftarrow{\mathcal{R}} \in \{0,1\}$ **Output** : $\mathsf{Enc}[\delta]$

USER OUTSOURCED LOCATION

14 : **get** $c_i, \mathsf{Enc}(\varphi)_{j_3} \leftarrow_\$ C$ *from* **User**
15 : **for** $j \leftarrow_\$ 1 \cdots m$ **do**
16 : **for** $i \leftarrow_\$ 1 \cdots n$ **do**
17 : $\mathsf{Enc}(\varphi)_{j_1} \leftarrow_\$ \mathsf{Enc}\left(\sum_{i=1}^{n} V_{ij}^2\right)$
18 : $\mathsf{Enc}(\varphi)_{j_2} \leftarrow_\$ \prod_{i=1}^{n} \left(c_i\right)^{v_{i,j}}$
19 : $\mathsf{Enc}(\delta_j) \leftarrow_\$ \mathsf{Enc}(\varphi)_{j_1} \cdot \mathsf{Enc}(\varphi)_{j_2} \cdot \mathsf{Enc}(\varphi)_{j_3}$
20 : **return** $\mathsf{Enc}(\delta) \leftarrow_\$ \left\{\mathsf{Enc}(\delta_j)\right\}_{j=1}^{m}$

Step (2) *of Protocol* 1

MASKING THE DISTANCES

21 : **for** $j \leftarrow_\$ 1 \cdots m$ **do**
22 : $\mathsf{Enc}(\gamma_j) \leftarrow_\$ \mathsf{Enc}(pk_s, \gamma_j)$
23 : $\mathsf{Enc}[\delta]_j \leftarrow_\$ \mathsf{Enc}(\delta_j) \cdot \mathsf{Enc}(\gamma_j)$
24 : **return** $\mathsf{Enc}[\delta] \leftarrow_\$ \left\{\mathsf{Enc}[\delta]_j\right\}_{j=1}^{m}$
25 : **post** $\mathsf{Enc}[\delta]$ *to* **CS**

is described in Algorithm 3.

$$[\delta]_j = \delta_j + \gamma_j \tag{13}$$

In the first part, which is to remove the noise (γ), the SP and the CS use Yao's protocol (garbled circuit) and OT. The SP constructs a garbled circuit by first representing a subtraction function $a^{\hat{l}} - b^{\hat{l}} = [\delta] - \gamma$ as a Boolean circuit, where $\hat{l}$ is the bit-length of the distance. Substituting equation 13 into this function yields:

$$a - b = [\delta] - \gamma = (\delta + \gamma) - \gamma = \delta \tag{14}$$

The subtraction circuit is similar to the one in [39] and [40], takes 2-input AND gate and 'free' XOR [32] computations (i.e., evaluation of the XOR-gates do not require any garble tables). The second part of the garbled function, $\mathcal{F}$, is to perform the comparisons & conditional swap using the technique of [6]. The entire circuit is converted into a garbled circuit (encrypted circuit) and sent to CS. The SP's input values of the circuit $\gamma_1, \cdots, \gamma_m$, are converted into garbled values $\tilde{\gamma}_1, \cdots, \tilde{\gamma}_m$, which can be sent along with the circuit to CS. The garbled input of CS, $\left[\tilde{\delta}_1\right], \cdots, \left[\tilde{\delta}_m\right]$ corresponding to the masked distances $[\delta_1], \cdots, [\delta_m]$ are transferred to CS using parallel OT to enable the CS evaluates the circuit as shown in step (3) of Algorithm 3. At the end of the evaluation, CS obtains the indexes $\pi_1, \cdots, \pi_k$ corresponding to the k-smallest Euclidean distances $\delta_1, \cdots, \delta_k$. The CS then forwards the output indexes $\{\pi_i\}_{i=1}^{k}$ to the mobile user (line 39 of algorithm III); these indexes are indistinguishable from random noise by the CS without the translation output table/function.

4) Retrieval Phase: When the user receives the garbled output indexes $\{\pi_i\}_{i=1}^{k}$ of the smallest k-nearest Euclidean distances, she uses the output function/translation table $\left[(\mathcal{H}(\pi^0), 0), (\mathcal{H}(\pi^1), 1)\right]$ sent by the SP to translate the garbled values (garbled output) to the corresponding plain indexes. The plain k-indexes correspond to the k-smallest euclidean distances d_j as depicted in step (4) of Algorithm 1. With the translation table, the user can determine the semantics of the output wire, i.e., $\mathcal{H}(\pi^0)$ represents 0 and $\mathcal{H}(\pi^1)$ represents 1 for all the $\hat{l}$ bits of each of the received euclidean distances. The user then estimates the private location d by computing the centroid of the k-nearest neighbours using $\mathcal{T}_2$ (the list of reference points).

TABLE III
COMPARISON OF COMPUTATIONAL COMPLEXITY AT USER-SIDE

Protocol	Preprocessing	Retrieval
[2]	$2n\ Exp + 4n\ \ Mul$	$m\ Exp$
[1]	$2n\ Exp + 4n\ \ Mul$	$m\ Exp$
This protocol	$(n+1)\ Exp$	$negligible$

5) Server Localization Phase: When the SP makes a request about the user's location, the user determines the geocode of $d(x_i, y_i) \in \mathcal{T}_2$ (list of reference points). Using a keyed hash function $\mathcal{H}^\star$, the user computes the $\mathcal{H}^\star(d)$, i.e., the digest over the location d, and sends the results $\{h_1(d), \cdots, h_k(d)\}$ to the CS, where $h_i(d)$ is the hash values/digest over d.

When CS receives the hash values $\{h_i(d)\}_{i=1}^{k}$, CS creates a Bloom filter B_u, by mapping the hash values unto the filter and the number of $1s = \alpha$ counted since this filter is binary and only takes 1s and 0s as input. Due to the multiplicative homomorphic properties, CS can compute $\mathsf{Enc}(B_u^{\#}) = \mathsf{Enc}(B^{\#}) \boxdot B_u$, using SP's public key (pk_s). If the user's current location was mapped unto to the spatial Bloom filter (SBF) by the SP, then after the multiplication of the two filters, all non-zero values in $\mathsf{Enc}(B_u^{\#})$ corresponding to the user's mappings in filter B_u, will remain, while all other positions in $\mathsf{Enc}(B_u^{\#})$ become zero. Finally, CS permutes the results, $\mathsf{Enc}(B_u^{\#})$, which changes the order to hide the location

Algorithm 3 Cloud Server (CS)

Output : π

OBTAINING USER'S EUCLIDEAN DISTANCES

get $\mathsf{Enc}[\delta]$ *from* **SP**
for $j \leftarrow_{\$} 1 \cdots m$ **do**
$[\delta_j] \leftarrow_{\$} \mathsf{Dec}(sk_u, \mathsf{Enc}[\delta_j])$
return $[\delta] \leftarrow_{\$} [\delta_1] \cdots [\delta_m]$

1-2-OBLIVIOUS TRANSFER

get $\left\{\mathcal{K}_i^0, \mathcal{K}_i^1\right\}_{i=1}^{m} \leftarrow_{\$} \mathcal{K}$ *from* **SP**
$[\delta_1] \cdots [\delta_m] \leftarrow_{\$} [\delta]$
for $j \leftarrow_{\$} 1 \cdots m$
  foreach $[\delta]$ **do**
  $\mathcal{K}_j^{\delta} \leftarrow_{\$} \{\mathcal{K}_1^{\delta_1}, \cdots, \mathcal{K}_m^{\delta_m}\}$
return $\mathcal{K}^{\delta} \leftarrow_{\$} \mathcal{K}_1^{\delta} \cdots \mathcal{K}_m^{\delta}$

} *Step* (3) *of Protocol* 1

CIRCUIT EVALUATION

for $j \leftarrow_{\$} 1 \cdots k$ **do**
  $\pi_i \leftarrow_{\$} \mathcal{F}(\mathcal{K}^{\delta})$
return $\pi \leftarrow_{\$} \pi_1 \cdots \pi_k$
post π *to* **User**

} *Step* (4) *of Protocol* 1

of the user, and sends both α and $\mathsf{Enc}(B_u^{\#})$ to the SP for verification.

When the SP receives the encrypted filter $\mathsf{Enc}(B_u^{\#})$, it decrypts it using the secret key sk_s and subsequently performs location verification. If the number of non-zeros in the filter is $< \alpha$, then certainly the user is not within an area of interest, i.e., $d \notin \nabla$. Else, if the number of non-zeros equals α, then the smallest output value ∇_j, will be the area that contains the geocode of the user's location, i.e., $(d \in \nabla_j)$, minus the false probability.

IV. Security

The security of the proposed Protocol 1 relies on the underlying cryptographic primitives used. The security model is based on a malicious user, and honest-but-curious non-colluding service provider (SP) and cloud server (CS). In other words, the SP and CS can analyze the data they receive to try to recover other parties' inputs, but should not deviate from the protocol, and should not be under the same authority/administrator. The user can instead provide arbitrary inputs and deviate from the protocol. This is novel with respect to the leading solutions in the literature, including [1] and [2], which offer protection only against *honest-but-curious* adversaries.

This security model, and in particular the non-collusion requirement, is established in the literature and is in line with current research on outsourced multi-party computation [41], [42], [43]. We argue that the model is also in line with industry practice and real-world expectations. While it is appropriate to assume a service provider and outsourced cloud server could become "curious", that is, learn as much as possible from their interactions, it is also realistic to assume they do not actively deviate from the protocol. This, however, can not be assumed of a user. Non-collusion is also a common legal requirement in the industry sector, for instance in the case of financial and technical independent audits [44].

In the following, we first provide an informal overview of the privacy achieved by the protocol (Section IV-A); then we provide a formal security definition (Section IV-B) and proof (Section IV-C).

A. Privacy

Privacy in this protocol is defined as follows:

- The SP learns only if the user is in a predefined area of interest ∇, not the exact location. If the user is outside these areas, the SP remains unaware of the user's location.
- The user learns only their estimated private location based on their input, without gaining knowledge of the SP's reference database or the predefined areas (∇).
- The cloud server (CS) remains oblivious to both the user's location and the SP's database.

The user's input (V_u) is encrypted using her public key and the DGK-cryptosystem [21], which is semantically secure. This ensures the server cannot recover the user's plain RSS input within polynomial time without the secret key (sk_u).

The protocol settings, including the use of Garbled Circuits (GC) and Oblivious Transfer (OT), are designed to protect the privacy of both the user's location and the SP's database. The CS, despite having access to the user's private key sk_u, cannot deduce information from the masked Euclidean distances $[\delta]$ without the user's plain input.

The user's exact location is further protected by mapping hash values into a filter, which the SP cannot reverse due to the cryptographic nature of hash functions and the shuffling of the filter $\mathsf{Enc}(B_u^{\#})$. The SP can only infer if the user is within a predefined area of interest (∇), without pinpointing the exact location. Though there might be some information leakage regarding the location of the user to the CS, should the user visit the same location more than once: this would be evident to the CS, which would however remain oblivious to the actual location. In other words, CS may learn that a user is visiting again a previous location, but would not learn the actual position of that location of the user. However, this can easily be mitigated by using session keys in the hash function computation, so that the same location would result in a different hash value for each session. This session-based mechanism ensures that the hash values are unpredictable and unique to each interaction.

From the SP's perspective, the user only receives output based on her input and does not participate in the computation after submitting her encrypted input. This limits the potential for a malicious user to corrupt the computation of her location.

The use of homomorphic encryption to encrypt the filter preserves the privacy of the predefined areas encoded in the filter, allowing the CS to compute privately without gaining additional information about $\mathsf{Enc}(B_u^{\#})$.

Despite potential adversarial attempts to construct their own database ($\tilde{\mathbb{D}}$), this is either impractical or ineffective due to controlled environments and the high accuracy of the SP's

well-trained reference database ($\mathbb{D}$). Attacks on the physical infrastructure are beyond the scope of this work.

B. Security Definition

We formulate a security definition based on the Real/Ideal world paradigm similar to [45], [46], and [47]. As indicated earlier, our model setting involves 3-parties in the execution of the protocol. One of the parties (the user, such as a mobile user) can be malicious. The other two servers (SP and CS) are independent non-colluding parties. In this setting, each party's input is simulated in an ideal view of the execution of the protocol. And, if the view during the simulation in the ideal world is computationally indistinguishable from that of the real execution of the protocol (Λ) based on the party's input/output, then the protocol is considered secure. In other words, no party learns any additional information except what the protocol intended to output. Furthermore, our secure multiparty computations protocol has only one-output function $f(x_1, x_2)$ based on the user's and SP's input. We now define the Ideal/Real-world execution of the protocol below:

Real-World Execution: The execution of the protocol (Λ) in the real-world model involves parties $P = \{U, CS, SP\}$ and a corresponding subset adversary $A = \{A_U, A_{CS}, A_{SP}\}$ who can corrupt the associated party. Let x_i be the protocol input, and $b_i \in_R \{0, 1\}^*$ be the auxiliary input (a string or side information that an attacker may have prior to the protocol's execution) and k the security parameter during the execution of the protocol. Except for the CS, who only inputs b, all parties input x and b. The two servers (SP and CS) are semi-honest and non-colluding, according to the model assumption. As a result, both servers will correctly input the appropriate data. This condition, however, does not apply to the malicious user. Let OUT^i be the output for the party $A_i \in A$, and OUT^j for $i \neq j$, be the output of the honest party during the execution of the protocol. Thus the protocol output during the real execution (*real view*) of the protocol is defined as:

$$REAL_{\Lambda}^{View}[A](\bar{x}, k)_{k\in\mathbb{N}} \stackrel{\text{def}}{=} OUT_{\Lambda}^{i} \cup \{OUT_{\Lambda}^{j} : j \in H\}$$

where $\bar{x} = \{x_1, x_2, b_1, b_2, b_3\}$, and H is honest parties.

Ideal-World Execution: The ideal world of the protocol execution has the same settings as the Real-world model with the same parties and input. However, in the ideal world, a trusted party receives all input from the parties participating in the protocol. The trusted party (TTP) securely computes the output function based on the received parties' input.

Thus let $f : \{0, 1\}^* \times \{0, 1\}^* \rightarrow \{0, 1\}^*$, represent the ideal functionality where:

Inputs: x_i denotes the input of User (U) and Service provider,(SP). Let b_i be the auxilliary input of adversary A and security parameter k for all party (U, CS,SP) including TTP.

The honest party (semi-honest) P_j always sends its input ($x_i' = x$) as required to the trusted party (TTP). However, the corrupted party A_i (malicious) can decide whether to send its arbitrary input (x') or not to the TTP. Should the corrupted party decide not to send its input, an empty string ($\perp$) or abort message is sent to the trusted party. In that case, an abort message is delivered to all parties. Else, the execution of the protocol continues, i.e., the trusted party (TTP) evaluates the function f.

Output: Let OUT^j be the output ($f(x_1', x_2')$), returned to the honest party $H \subseteq P$ by the trusted party (TTP) and OUT^i be the arbitrary output (computed within probalistic polynomial time) returned to the corrupted party A_i. Thus the i-th partial output of the *ideal-world* execution of f between the parties (U, CS, SP) in the presence of adversary A (independently malicious simulators) is given by:

$$IDEAL_{f}^{View}[A](x_1, x_2, k)_{k\in\mathbb{N}} \stackrel{\text{def}}{=} OUT_{f}^{i} \cup \{OUT_{f}^{j} : j \in H\}$$

We now provide a security definition for the three-party protocol based on the introduction of the Ideal/Real-world protocol executions.

Definition 1 (Security): A three-party protocol Λ, securely computes the function f (with an abort option in the presence of malicious parties), if for every probabilistic polynomial time (PPT) adversary A, for the real model, $\exists$ a PPT adversary S in the ideal model, such that:

$$\left\{IDEAL_{f}^{View}[S, A](x_1, x_2, k) \stackrel{c}{\approx} REAL_{\Lambda}^{View}[A](\bar{x}, k)\right\}_{k\in\mathbb{N}}$$

where $\stackrel{c}{\approx}$ means *computationally indistinguishable*. Our privacy-preserving protocol for indoor Wi-Fi-localization is secure according to Theorem 1 below and meets the security Definition 1. The proof can be found in the following (Section IV-C).

Theorem 1: Protocol 1 computes the function f securely in the presence of a malicious user, a semi-honest SP, and a semi-honest CS.

C. Security Proof

In showing the protocol's security, we formulate the view of the real execution of the protocol by constructing the view of the simulator for each considered adversary (A_U, A_{CS}, A_{SP}) based on the input/output in the ideal world. If the view of the ideal world is computationally indistinguishable from the view in the real execution of the protocol, then we conclude that the protocol is secure. Thus, the adversary does not learn any extra information apart from what the protocol intended to output.

1) View of a Malicious User: From the user's perspective, its input to the protocol is the encryption of r_i, and the output is the garbled values of the k-nearest Euclidean distances. We choose a simulator S_U, which runs A_U, and controls the other parties. Similar to the real world, A_U obtains r_i (any arbitrary input) and sends it to S_U. As the adversary can make any arbitrary input, the experiment terminates if A_U sends an incorrect input bitlength or an empty string. Else, S_U simulates the user's view by encrypting r_i since it has the keys. Since the experiment controls both the SP and the CS, S_U computes the garbled values $\tilde{\gamma}_i$ and $\tilde{\delta}_i$ like in the real execution of the protocol Λ. S_U then queries the trusted party TP for the evaluation of $f(\tilde{\gamma}_i, [\tilde{\delta}_i])$. Finally, S_U sends to A_U the garbled output of the k-nearest Euclidean distances and the output translation function $\left[(\mathcal{H}(\pi^0), 0), (\mathcal{H}(\pi^1), 1)\right]$ as in the

real execution of the protocol. Due to the semantic security of the garbled output (random values), the view of the adversary A_U in the ideal world is indistinguishable from the view of the real-world execution of the protocol; that is, the ideal/real output of the protocol are computationally indistinguishable.

2) View of the (Semi-Honest) Service Provider: Now we construct a simulator S_{SP} for in the ideal world to simulate the adversary's (A_{SP}) view in a real execution of the protocol. In this scenario, S_{SP} runs A_{SP} and controls the other parties. In the ideal execution, the simulator S_{SP} prepares its input of choice, similar to the mobile user's input (r_i), in the real execution of the protocol. This input is encrypted and sent to the A_{SP}. The adversary A_{SP} now performs its computation similar to the steps in the real execution of the protocol and computes the garbled circuit as well. A_{SP} then sends to S_{SP} the garbled circuits alongside its garbled input $\tilde{\gamma}_i$. Since the adversary A_{SP} is semi-honest as, in the definition of the security model, the right input is expected to be sent else the experiment terminates.

Since the simulator controls the other parties; A_{SP} and the simulator engages in 1-2 OT protocol to deliver the keys corresponding to the simulator's input ($[\delta_i]$) as in the real execution of the protocol. The simulator S_{SP}, now having both input to the function f, queries the trusted party TP, to evaluate $f(\tilde{\gamma}_i, [\tilde{\delta}_i])$.

Due to the security of oblivious transfer, it is evident that the adversary A_{SP} cannot learn any information about the simulators' input of the garbled circuit. In addition, the semantic security (IND-CPA) of the DGK cryptosystem makes it impossible for A_{SP} to learn any meaningful information about the Mobile user's input within a polynomial time similar to the real execution of the protocol. Furthermore, any modification or incorrect construction of the circuit will cause the experiment to terminate, bearing in mind that the server is a semi-honest party this is not an issue. Thus, the adversary A_{SP} has no way to statistically distinguish the above interaction in the ideal world from, that of the real execution of the protocol.

3) View of the (Semi-Honest) Cloud Server: Finally, we construct a simulator (S_{CP}) which in the ideal world can simulate the adversary's view (A_{CP}) in the real execution of the protocol.

The output of CS is the l-bits of the smallest Euclidean distances (garbled output). In this experiment, the simulator S_{CS} runs A_{CS} and controls the other parties (SP and the user). First S_{CP}, sends to A_{CP} an encrypted masked distance ($\mathsf{Enc}[\delta]$). This allows A_{CP} to decrypt the encrypted, masked distance as in the real-execution of the protocol. The simulator S_{CP} then simulates a garbled circuit and sends it to A_{CS} alongside the server's garbled input ($\tilde{\gamma}_i$). Finally S_{CP} simulates an OT protocol, and engages A_{CP} in a 1-2 oblivious transfer protocol to allow A_{CS} to recover the keys to its garbled input. A_{CP} then evaluates the circuit and sends back to S_{CP} garbled output values of Euclidean distances. Since the A_{CP} is semi-honest per the definition of the security model, it will not tamper with the garbled output, else that will cause S_{CP} to trigger ($\perp$), guaranteeing the correct output of the garbled circuit. Also, the semantic security (IND-CPA) of the garbled output and the security of the Oblivious transfer protocol makes the garbled values indistinguishable from pseudo-random noise. As a result, the A_{CS} is unable to distinguish between the simulated interaction in the ideal world and the real execution of the protocol.

D. Potential Expansion to a Malicious Service Provider

Although this protocol was designed to provide security against a semi-honest service provider, in this section, we introduce and briefly discuss some potential capabilities when the service provider is assumed to be malicious. This might be useful for the reader in comprehending the reasoning behind the choice of the security model for the protocol construction. As a malicious adversary, the most likely scenario for the SP is to construct the Spatial Bloom filter so that verifying the user's location is always true. In other words, the SP will include all locations of the infrastructure in question so that the user's location is always disclosed, regardless of whether the user is within an area of interest. However, we believe that additional hardening of the protocol could easily reveal that the SP is cheating, as the size of the areas of interest could be agreed upon beforehand, revealing the size of the filter; thus, any additional insertion into the filter will result in the SP being caught, as the size of the filter will exceed what was agreed upon. Also, as SP is the constructor of the garbled circuit, in order to send the garbled version of **F**, SP can easily send the incorrect F to CP, as SP is not required to adhere to the protocol's instructions. As CS only receives ciphertext or encrypted circuits, it is unable to confirm that the correct **F** or circuit was sent. This intent will affect the protocol's correctness. In reality, this is a problem of both correctness and privacy. Thus, the output of the evaluation may appear to be random garbage, but it reveals the CS's entire input (user's location). A potential solution to address the issue of a malicious party constructing a circuit is to use the cut-and-choose paradigm. This approach involves the constructor creating multiple copies of the circuits and the evaluator requesting the constructor to open more than half of them. By doing so, if the constructor has constructed the circuit incorrectly, there is a high probability of being caught during the verification process. This approach provides a way to ensure the security and accuracy of the circuit construction in the presence of malicious parties.

V. Evaluation

This work makes use of homomorphic encryption in preserving the privacy of the users' queries. Using homomorphism clearly increases the computation and communication incurred on the user side, which is why in the design of the protocol, we outsourced most of the user's computations. Due to the higher bandwidth and computational capabilities, we can safely assume to be at the disposal of the SP and CS, the cost analysis focuses more on the user side, which we presume to be a constrained mobile or IoT device. For completeness, we nonetheless include the SP and CS computation and communication costs in Section V-C.

For this analysis, we use the recommendations [48], [49] of key size up-to-long-term protection. The security parameters;

T denotes the asymmetric key and t for the symmetric key. Though a key-length $T = 1024$ (in Table V) is widely acknowledged to be obsolete for security by the cryptographic community, we included it for the sake of completeness of the analysis.

In the following, we evaluate the computational and communication complexity of the proposed cryptographic protocol, and we compare it to the current leading solutions in the literature. In particular, we compare it directly to [1] and [2], as these schemes have comparable constructions. As neither of these schemes has been implemented, we compare the computational complexity in terms of exponentiations and multiplications, as standard in the relevant literature. In both cases, the proposed protocol has a significant complexity advantage. This stands even if the improvements to previous protocols proposed in [11] for [2] are included.

A. Computational Complexity

Analyzing computational complexity is essential to secure multi-party computation (SMC) efficiency, especially considering IoT devices.

Localization protocols using additively homomorphic encryption mostly use Paillier encryption [25], where the hardness relies on the composite residual problem with RSA modulus N as the public key. This protocol instead uses DGK [21], [22], which has smaller plaintext space but with better performances. We use the DGK for encryption while the GC is for the subtraction and computing of the shortest Euclidean distances.

To make fair comparisons with other similar localization protocols, we use the optimized version of these protocols where most of the expensive modular exponentiation is pushed to the set-up phase. Table III depicts a summary of the findings. In protocols [1], [2], a total of $2n$ elements (RSS) of the user's input query are encrypted using the Paillier encryption. The cost of Paillier encryption requires two modular exponentiation (Exp) and one modular multiplication (Mul). The complexity of the encryption is improved by choosing $g = N + 1$ in the equation 2 to become:

$$\mathsf{Enc}(x) = (xN + 1) \cdot r^N \mod N^2 \tag{15}$$

making the encryption complexity to be only one exponentiation and two modular multiplications instead. Thus in total, each user requires $2n$ exponentiation and $4n$ modular multiplications for each input query. Using the DGK cryptosystem results in the encryption complexity of one exponentiation modulo N being more efficient than the Paillier due to the lower exponent. In total, the user requires approximately n exponentiation which is a significant improvement. In [2] and [1], decrypting one ciphertext requires one exponentiation with optimization technique while using our outsourced protocol requires negligible computation.

Comparing this protocol to [11], which suggested the use of Paillier and GC as a solution to the security weakness in [2], our proposed protocol has better performance, especially the heavy GC computations at the user side are all delegated to the CS, making it more practicable in a real-world setting.

TABLE IV

TIME REQUIRED BY THE USER TO PERFORM THE REQUIRED ENCRYPTION COMPUTATION, FOR DIFFERENT KEY SIZES. THE TABLE DETAILS THE MEAN ENCRYPTION TIMES IN MILLISECONDS (MS) AVERAGED OVER 10 ENCRYPTION ROUNDS. THIS ILLUSTRATES THE RELATIONSHIP BETWEEN INCREASING KEY SIZE AND ENCRYPTION DURATION, HIGHLIGHTING THE COMPUTATIONAL IMPACT OF STRONGER ENCRYPTION IN SECURE COMMUNICATIONS

Key Size (bits)	Encryption Time (ms)
1024	0.14
2048	0.38
3072	0.79
4096	1.26

Even when most of the computations are in the setup phase, users in [11] in the online phase still need to compute n exponentiation for n ciphertexts and GC evaluation requiring lm non-XOR gates for subtraction. Evaluating the GC for the indexes of k smallest distances by the user using the k nearest neighbour search algorithm of [50], and later improved by [6] would still require the invocation of cryptographic hash functions of $k(2ml + m \log_2 m)$ non-XOR gates for k comparisons and $2k$ conditional swap, where $\log_2 m$ is the circuit depth for m values of bit-length $\hat{l}$. This computation is undoubtedly expensive for a mobile device (especially IoT) with limited resources to perform compared to the proposed protocol.

To assess the computational burden on the user-side, particularly for devices with constrained processing power, we provide experimental results on the time required to perform the encryptions needed to securely transmit data to the servers. The analysis focuses on the user, as SP and CS are presumed to be equipped with robust computational resources. The implementation was done using Python, and run on a 2.40 GHz Intel(R) Core(TM) processor. The results, illustrated in Table IV, comprise the time complexity for both equations 7a and 7b, which are encryptions performed by the user before transmitting data to the Service provider (SP). Specifically, we use a sample $r_i = \{-40, -42, -46, -49, -50, -55\}$, obtained from an experiment on live access points. This evaluation is crucial for understanding the practicality and efficiency of the protocol in real-world scenarios, especially when used by devices with constrained resources. The results highlight the relatively small computational effort and time required, which would be easily achieved on any modern user device (such as a smartphone).

B. Communication Complexity

The communication complexity of Paillier and DGK is largely dependent on the size T of the RSA modulus N. The size of the ciphertext of the Paillier cryptosystem is as twice as that of DGK. Moreover, unlike protocol [1], [2], where $2n$ output ciphertexts are sent from the user to the SP, the user in this protocol sends a total of $n + 1$ ciphertexts to the SP, an improvement by a factor of approximately 2 in the online communication.

TABLE V
COMPARISON OF BANDWIDTH USAGE AT USER-SIDE: FOR THIS PROTOCOL, THE USER'S TRAFFIC EMANATES FROM THE CLOUD SERVER

	Preprocessing			Online		
Security Level	Related work [2]	Related work [1]	This Protocol	Related work [2]	Related work [1]	This Protocol
Asymptotic	$4nT$	$4nT$	$(n+1)T$	$2mT$	$2mT$	$k\hat{l}$
1024	6.1 kB	6.1 kB	1.7 kB	236.5 kB	236.5 kB	0.015 kB
2048	12.3 kB	12.3 kB	3.3 kB	473.1 kB	473.1 kB	0.015 kB
3072	18.4 kB	18.4 kB	5.0 kB	709.6 kB	709.6 kB	0.015 kB
15360	92.1 kB	92.1 kB	25.0 kB	3548.2 kB	3548.2 kB	0.015 kB

TABLE VI
COMPUTATIONAL AND COMMUNICATION COMPLEXITY AT THE SP AND CS

		Preprocessing			Online		
		DGK	GC	OT			
Computation	SP	$mn\ Exp$	$2kt(2m\hat{l}+m\log_2 m)$	$2\hat{l}$	$\mathcal{L}m\ Exp+m(n+3)Mul+km$		
	CS		$kt(2m\hat{l}+m\log_2 m)$	$\hat{l}$	$m\ Exp+1\ Mul$		
Communication*			$3\hat{l}mt$	$6\hat{l}mt$	DGK mT	GC $\hat{l}mt$	OT $2m\hat{l}t$

* The total communication between the SP and the CS.

Table V shows a significant gain in this protocol's communication bandwidth compared to that of [1] and [2]. The user's encrypted message sent to the SP in this protocol has a communication bit of $(n+1)T$ asymptotically compared to $4nT$ of [1], [2]. As shown in Table V as the keysize increases, the gain in bandwidth in our protocol asymptotically approaches a factor of 4.

The encrypted distances that the SP transfers to the user in [1] and [2] require an asymptotic communication overhead of $2mT$ bits; juxtaposing it, this protocol has a much lower communication complexity of kl bits. Table V depicts the pattern of gains for different security parameters. Let $m = 924$ and $n = 520$ in the database ($\mathbb{D}$) as in [51]. The scan result at each reference point depends on the internal structure and the positions of the installed access points. Reference [2] estimated an average of 6 APs is enough to localize a user. To estimate the communication in bandwidth, we set $n = 12$[1] *goal* in table V for the average scan result per reference point that a user sends to the SP. And let $k = 5$ and maximum of $\hat{l} = 24$, (where $\hat{l} = log_2 n + 2l\ bits + 1$, is the maximum bit-length of the euclidean distance). During the online phase, as the key-size of [1] and [2] increases, the bandwidth size grows proportionally while in our proposed protocol, the bandwidth remains constant.

The estimated communication cost in the suggested solution [11] in the user/server settings still poses a significant communication overhead at the user side. The communication cost for sending one Paillier ciphertext will be $2T$. The OT protocol of [53] against a semi-honest SP and the malicious user will result in communication complexity bits of $\approx 12mlt$. And the offline communication complexity for transferring the GC will be lm non-XOR gates for the subtraction and approximately $3klm$ (klm for comparisons and $2klm$ for the minimum selection) for computing the Euclidean distances. Thus, using the Garbled row reduction of [35] will result in a complexity of approximately $3t$ bits per non-XOR gate.

Therefore, this outsourced protocol brings significant communication bandwidth gains, considering the limited resources on the user side. Though combining the efficient OT protocol of [28] and [30] will allow most of the expensive operations to be done in the set-up phase, and in effect, improve the complexity. Nevertheless, the bandwidth involved would still be an overburden for a resource constraint user in addition to the direct impact on power consumption. Without the outsourcing server, the total online communication complexity in the phase of optimization would asymptotically be $2mT$ for m Paillier ciphertexts transferred by the SP plus $2m\hat{l}t$ of the online part of the OT protocol for m values of bit-length $\hat{l}$ and security parameter t. Thus communication complexity of $k\hat{l} - bits$ for k values of bit-length $\hat{l}$ in this protocol for resource constraint mobile user is a significant improvement over [11], not to mention the communication overhead in the set-up phase for the user.

C. Communication and Computation for SP and CS

Our focus for the cost analysis is mainly on the user side, who normally uses a constrained mobile device and is the party that benefits the most from a reduction in overheads. However, for completeness, we briefly discuss the approximate computational complexity of the service provider (SP) and cloud server (CS), as well as their communication costs. The results are summarised in Table VI, where $\mathcal{L} = 2^l - 1$ and l is the bit-length of the RSS value. To accurately track the cost of communication between a Service Provider (SP) and a Cloud Server (CS) in a secure computing environment, it is necessary to consider all relevant factors. Therefore, the communication Table VI records the full cost of this communication, which includes the DGK ciphertext, the total ciphertext sent in the Oblivious Transfer (OT) step, and the garbled values received by the CS. This differs from the first Table V, which merely monitors the total number of packets the user receives from the Cloud server. Considering these variables, the communication

[1]The work of [2] showed that scan results (number of available ap) ≥ 6 per reference point is enough to localize a user. In [51], the average number of available AP per reference point is 18 while in [52], it ranges from 11 to 67.

table provides a more comprehensive and realistic depiction of communication costs in a safe computing environment. It is important to remember that both parties (CS and SP) are very likely to have no constrain in communication capabilities and a very high computational capacity. Nonetheless, our results show that both computation and communication overheads are perfectly manageable even by low-resource servers.

In the set-up phase, a total of mn exponentiation is needed to compute $\mathsf{Enc}(v^2)$ plus m encryption of the noise (γ). For the garbled circuit, SP will invoke 2^2 hash functions for each 2-input non-xor gate created; the technique of [37] will bring it down to only 2 per 2-input non-xor gate. The random oracle (RO) protocol of [30] will result in $2\hat{l}$ invocation of cryptographic hash functions for the computation of OT by the SP in the set-up phase. In computing the filter, the SP will invoke approximately k cryptographic hash functions for each position inserted in the filter. The CS, on the other hand, only needs one invocation of the cryptographic hash function for each 2-input non-xor gate. It is important to highlight that most SP's computations would be more cost-effective than shown in the theoretical analysis, as most RSS values are zero at each reference point. Given that cloud service providers have state-of-the-art infrastructure and tremendous resources at their disposal, computational work by the CS is easily doable. Computations for all non-xor gates are negligible.

In the online phase, the cost for computing $\mathsf{Enc}(\varphi)_{j_2}$ will be $\approx \mathcal{L}m$ exponentiations (where $\mathcal{L} = 2^l - 1$ and l is the bit-length of the RSS value) plus mn modular multiplications. Finally, $2m$ modular multiplications are required in computing $\mathsf{Enc}(\delta)$ and m modular multiplication for computing $\mathsf{Enc}[\delta]$.

With regards to communication complexity, the SP will send a total of $3\hat{l}t$ per each 2-input non-XOR gates using the garbled row reduction [35] for the garbled circuit plus $\approx 6\hat{l}t$ for the OT based on the random oracle [30]. In the online phase, the communication cost will be asymptotically mT of DGK ciphertext and $3\hat{l}mt$ *bits* ($\hat{l}mt$ for SP's garbled input and $2\hat{l}mt$ for the OT based on two rounds of communication between SP and CS).

VI. Conclusion

In this paper, we propose an efficient and privacy-preserving protocol for indoor Wi-Fi localization that guarantees both the service provider and the user's privacy. Our protocol combines a number of cryptographic primitives to preserve the privacy of the parties involved. In particular, our scheme employs additive homomorphic encryption (DGK encryption) to preserve the privacy of the user's location fingerprint while allowing the service provider to make computations over the encrypted fingerprint. Whereas the use of a garbled circuit preserves the privacy of the SP's reference database against malicious users, and at the same time, delivers the location output (k-nearest neighbours) to the user. The spatial Bloom filter allows the SP to use the location-based service in the indoor environment to learn the user's vicinity in predefined areas of interest while preventing the user from learning these predefined areas. When the user happens to be outside these predefined areas of interest (∇), the service provider learns nothing about the user's location.

Compared to other similar protocols [1], [2], our protocol's complexity analysis shows a very significant reduction in computational costs at the user-side, as most of the heavy computations of the garbled circuit operations are securely and effectively outsourced to the cloud server. Online communication costs show a reduction of about 99.99% compared to other similar protocols in the user/server setting, making our protocol more efficient and practicable in the Internet of Things environment. Additionally, our protocol is the first among those analyzed to provide security against malicious users, with other protocols limited to honest-but-curious adversaries.

Potential directions for future work include extending the protection against malicious cloud servers/service providers and implementing it by leveraging the ABY [54] framework for mixed protocol multi-party computation.

References

[1] S. N. Eshun and P. Palmieri, "A privacy-preserving protocol for indoor Wi-Fi localization," in *Proc. 16th ACM Int. Conf. Comput. Frontiers*, Apr. 2019, pp. 380–385.

[2] H. Li, L. Sun, H. Zhu, X. Lu, and X. Cheng, "Achieving privacy preservation in WiFi fingerprint-based localization," in *Proc. IEEE Conf. Comput. Commun. (INFOCOM)*, Apr. 2014, pp. 2337–2345.

[3] A. Göker and H. I. Myrhaug, "User context and personalisation," in *Proc. 6th Eur. Conf. Case Based Reasoning*, 2002, pp. 1–7.

[4] NOAA. (2022). *GPS Accuracy*. [Online]. Available: https://www.gps.gov/systems/gps/performance/accuracy/

[5] J. Torres-Sospedra, R. Montoliu, S. Trilles, Ó. Belmonte, and J. Huerta, "Comprehensive analysis of distance and similarity measures for Wi-Fi fingerprinting indoor positioning systems," *Expert Syst. Appl.*, vol. 42, no. 23, pp. 9263–9278, Dec. 2015.

[6] K. Järvinen et al., "PILOT: Practical privacy-preserving indoor localization using OuTsourcing," in *Proc. IEEE Eur. Symp. Secur. Privacy*, Jun. 2019, pp. 448–463.

[7] I. Oleynikov, E. Pagnin, and A. Sabelfeld, "Outsourcing MPC precomputation for location privacy," in *Proc. IEEE Eur. Symp. Secur. Privacy Workshops*, Genoa, Italy, Jun. 2022, pp. 504–513.

[8] P. Zhang et al., "Privacy-preserving and outsourced multi-party K-means clustering based on multi-key fully homomorphic encryption," *IEEE Trans. Dependable Secure Comput.*, vol. 20, pp. 2348–2359, May/Jun. 2023.

[9] L. Calderoni, M. Ferrara, A. Franco, and D. Maio, "Indoor localization in a hospital environment using random forest classifiers," *Expert Syst. Appl.*, vol. 42, no. 1, pp. 125–134, Jan. 2015.

[10] T. Araki, J. Furukawa, Y. Lindell, A. Nof, and K. Ohara, "High-throughput semi-honest secure three-party computation with an honest majority," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.*, Oct. 2016, pp. 805–817.

[11] Z. Yang and K. Järvinen, "The death and rebirth of privacy-preserving WiFi fingerprint localization with Paillier encryption," in *Proc. IEEE Conf. Comput. Commun.*, Apr. 2018, pp. 1223–1231.

[12] T. Zhang, S. S. M. Chow, Z. Zhou, and M. Li, "Privacy-preserving Wi-Fi fingerprinting indoor localization," in *Proc. 11th Intl. Workshop Secur.*, 2016, pp. 215–233.

[13] T. Shu, Y. Chen, J. Yang, and A. Williams, "Multi-lateral privacy-preserving localization in pervasive environments," in *Proc. IEEE Conf. Comput. Commun.*, Apr. 2014, pp. 2319–2327.

[14] C. Yang, Z. Jia, and S. Li, "Privacy-preserving proximity detection framework for location-based services," in *Proc. Int. Conf. Netw. Netw. Appl. (NaNA)*, Oct. 2021, pp. 99–106.

[15] M. Raykova, B. Vo, S. M. Bellovin, and T. Malkin, "Secure anonymous database search," in *Proc. ACM Workshop Cloud Comput. Secur.*, Chicago, IL, USA, Nov. 2009, pp. 115–126.

[16] E. Stefanov and E. Shi, "Multi-cloud oblivious storage," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.*, Berlin, Germany, 2013, pp. 247–258.

[17] S. Kamara, P. Mohassel, M. Raykova, and S. S. Sadeghian, "Scaling private set intersection to billion-element sets," in *Proc. 18th Int. Conf. Financial Cryptogr. Data Secur.*, vol. 8437. Cham, Switzerland: Springer, 2014, pp. 195–215.

[18] P. Palmieri, L. Calderoni, and D. Maio, "Spatial Bloom filters: Enabling privacy in location-aware applications," in *Proc. 10th Int. Conf. Inf. Secur. Cryptol.*, 2014, pp. 16–36.
[19] L. Calderoni, P. Palmieri, and D. Maio, "Location privacy without mutual trust: The spatial Bloom filter," *Comput. Commun.*, vol. 68, pp. 4–16, Sep. 2015.
[20] L. Calderoni, P. Palmieri, and D. Maio, "Probabilistic properties of the spatial Bloom filters and their relevance to cryptographic protocols," *IEEE Trans. Inf. Forensics Security*, vol. 13, no. 7, pp. 1710–1721, Jul. 2018.
[21] I. Damgård, M. Geisler, and M. Krøigaard, "Efficient and secure comparison for on-line auctions," in *Proc. 12th Australas. Conf. Inf. Secur. Privacy*, 2007, pp. 416–430.
[22] I. Damgard, M. Geisler, and M. Kroigard, "A correction to 'efficient and secure comparison for on-line auctions,'" *Int. J. Appl. Cryptogr.*, vol. 1, no. 4, p. 323, 2009.
[23] A. C. Yao, "Protocols for secure computations (extended abstract)," in *Proc. 23rd Annu. Symp. Found. Comput. Sci.*, 1982, pp. 160–164.
[24] A. C. Yao, "How to generate and exchange secrets (extended abstract)," in *Proc. 27th Annu. Symp. Found. Comput. Sci.*, 1986, pp. 162–167.
[25] P. Paillier, "Public-key cryptosystems based on composite degree residuosity classes," in *Proc. Int. Conf. Theory Appl. Cryptograph. Techn.*, Jan. 1999, pp. 223–238.
[26] S. C. Pohlig and M. E. Hellman, "An improved algorithm for computing logarithms over $GF(p)$ and its cryptographic significance (corresp.)," *IEEE Trans. Inf. Theory*, vol. IT-24, no. 1, pp. 106–110, Nov. 1978.
[27] M. Rabin, "How to exchange secrets by oblivious transfer," Harvard Aiken Comp. Lab, Cambridge, MA, USA, Tech. Rep. TR-81, 1981.
[28] D. Beaver, "Precomputing oblivious transfer," in *Proc. Annu. Int. Cryptol. Conf.*, 1995, pp. 97–109.
[29] M. Naor and B. Pinkas, "Computationally secure oblivious transfer," *J. Cryptol.*, vol. 18, no. 1, pp. 1–35, Jan. 2005.
[30] M. Naor and B. Pinkas, "Efficient oblivious transfer protocols," in *Proc. 12th Annu. ACM-SIAM Symp. Discrete Algorithms*, 2001, pp. 448–457.
[31] Y. Lindell and B. Pinkas, "A proof of security of Yao's protocol for two-party computation," *J. Cryptol.*, vol. 22, no. 2, pp. 161–188, Apr. 2009.
[32] V. Kolesnikov and T. Schneider, "Improved garbled circuit: Free XOR gates and applications," in *Proc. Int. Colloq. Automata, Lang., Program.*, vol. 5126, Jul. 2008, pp. 486–498.
[33] D. Beaver, S. Micali, and P. Rogaway, "The round complexity of secure protocols (extended abstract)," in *Proc. ACM Symp. Theory Comput. (STOC)*, 1990, pp. 503–513.
[34] D. Malkhi, N. Nisan, B. Pinkas, and Y. Sella, "Fairplay—A secure two-party computation system," in *Proc. 13th USENIX Secur. Symp.*, 2004, pp. 287–302.
[35] M. Naor, B. Pinkas, and R. Sumner, "Privacy preserving auctions and mechanism design," in *Proc. 1st ACM Conf. Electron. Commerce*, Nov. 1999, pp. 129–139.
[36] B. Pinkas, T. Schneider, N. P. Smart, and S. C. Williams, "Secure two-party computation is practical," in *Proc. Int. Conf. Theory Appl. Cryptol. Inf. Secur.*, vol. 5912, M. Matsui, Ed. Berlin, Germany: Springer, Dec. 2009, pp. 250–267.
[37] S. Zahur, M. Rosulek, and D. Evans, "Two halves make a whole—Reducing data transfer in garbled circuits using half gates," in *Proc. EUROCRYPT*, 2015, pp. 220–250.
[38] B. H. Bloom, "Space/time trade-offs in hash coding with allowable errors," *Commun. ACM*, vol. 13, no. 7, pp. 422–426, Jul. 1970.
[39] T. Schneider, *Engineering Secure Two-Party Computation Protocols—Design, Optimization, and Applications of Efficient Secure Function Evaluation.* Cham, Switzerland: Springer, 2012.
[40] V. Kolesnikov, A. Sadeghi, and T. Schneider, "Improved garbled circuit building blocks and applications to auctions and computing minima," in *Proc. 8th Int. Conf. Cryptol. Netw. Secur.*, 2009, pp. 1–20.
[41] H. Carter, B. Mood, P. Traynor, and K. Butler, "Secure outsourced garbled circuit evaluation for mobile devices," *J. Comput. Secur.*, vol. 24, no. 2, pp. 137–180, Apr. 2016.
[42] T. P. Jakobsen, J. B. Nielsen, and C. Orlandi, "A framework for outsourcing of secure computation," in *Proc. 6th ACM Workshop Cloud Comput. Secur.*, Nov. 2014, pp. 81–92.
[43] S. Kamara, P. Mohassel, and B. Riva, "Salus: A system for server-aided secure function evaluation," in *Proc. ACM Conf. Comput. Commun. Secur.*, Oct. 2012, pp. 797–808.
[44] F. Doelitzscher, C. Reich, and A. Sulistio, "Designing cloud services adhering to government privacy laws," in *Proc. 10th IEEE Int. Conf. Comput. Inf. Technol.*, Jun. 2010, pp. 930–935.
[45] O. Goldreich, *The Foundations of Cryptography—Volume 2: Basic Applications*, vol. 2. Cambridge, U.K.: Cambridge Univ. Press, 2004. [Online]. Available: http://www.wisdom.weizmann.ac.il/%7Eoded/foc-vol2.html
[46] S. Kamara, P. Mohassel, and M. Raykova, "Outsourcing multi-party computation," *IACR Cryptol. ePrint Arch.*, vol. 2011, p. 272, Oct. 2011. [Online]. Available: http://eprint.iacr.org/2011/272
[47] Y. Lindell, "How to simulate it—A tutorial on the simulation proof technique," in *Tutorials on the Foundations of Cryptography*. Cham, Switzerland: Springer, 2017, pp. 277–346, doi: 10.1007/978-3-319-57048-8.
[48] NIST. (2016). *Recommendation for Key Management, Special Publication 800–57 Part 1 Rev. 4*. [Online]. Available: https://csrc.nist.gov/Projects/Key-Management/publications
[49] ENCRYPT. (2018). *Algorithms, Key Size and Protocols Report (2018), H2020-ict-2014—Project 645421, D5.4, Ecrypt-CSA*. [Online]. Available: https://www.ecrypt.eu.org/csa/documents/D5.4-FinalAlgKeySizeProt.pdf
[50] E. M. Songhori, S. U. Hussain, A.-R. Sadeghi, and F. Koushanfar, "Compacting privacy-preserving k-nearest neighbor search using logic synthesis," in *Proc. 52nd ACM/EDAC/IEEE Design Autom. Conf. (DAC)*, Jun. 2015, pp. 1–6.
[51] J. Torres-Sospedra et al., "UJIIndoorLoc: A new multi-building and multi-floor database for WLAN fingerprint-based indoor localization problems," in *Proc. Int. Conf. Indoor Positioning Indoor Navigat. (IPIN)*, Oct. 2014, pp. 261–270.
[52] E. Lohan and J. Talvitie. (2014). *Indoor Wlan Measurement Data*. [Online]. Available: http://www.cs.tut.fi/tlt/pos/MEASUREMENTS_WLAN_FOR_WEB.zip
[53] W. Aiello, Y. Ishai, and O. Reingold, "Priced oblivious transfer: How to sell digital goods," in *Proc. EUROCRYPT*, 2001, pp. 119–135.
[54] D. Demmler, T. Schneider, and M. Zohner, "ABY—A framework for efficient mixed-protocol secure two-party computation," in *Proc. Netw. Distrib. Syst. Secur. Symp.*, 2015, pp. 497–511.

Samuel N. Eshun received the B.Sc. degree in mathematics and statistics from the University of Cape-Coast (UCC), Ghana, in 2011, the M.Sc. degree in coding theory and cryptography from the University of Trento, Italy, and the Ph.D. degree from University College Cork, Ireland. He is currently a Senior Cyber Security Engineer with Jaguar Land Rover, Shannon, Ireland. His current research interests include data privacy, cryptography, and secure multiparty computation, location privacy, and anonymization.

Paolo Palmieri (Member, IEEE) received the Ph.D. degree in cryptography from Université Catholique de Louvain, Belgium, in January 2013. Then, he was a Post-Doctoral Researcher with Delft University of Technology, The Netherlands, and also lectured with Bournemouth University and Cranfield University, U.K. He is currently a Lecturer in cyber security with University College Cork, Ireland. His research interests include cryptography, privacy, anonymity, secure computation, privacy-enhancing technologies, anonymity protocols, location privacy, and smart cities.