PPRP: Preserving Location Privacy for Range-Based Positioning in Mobile Networks

Cheng Huang, *Member, IEEE*, Dongxiao Liu, *Member, IEEE*, Anjia Yang, *Member, IEEE*, Rongxing Lu, *Fellow, IEEE*, and Xuemin Shen, *Fellow, IEEE*

***Abstract*—In this paper, we propose a privacy-preserving range-based positioning scheme, named PPRP, which can preserve the location privacy of both user equipment (UE) and anchors (ACs) in mobile networks. Specifically, PPRP is established on a decentralized trust-based framework that divides trust between two location management function (LMF) servers. With such a framework, UE and ACs are allowed to securely upload their range/range-difference measurement data to LMF servers using lightweight additive secret sharing techniques (ASS) instead of cumbersome cryptographic operations. Then, PPRP takes secret-shared measurement data as inputs and decomposes UE's location estimation procedures into secure two-party matrix computation sub-protocols, which are elaborately crafted using somewhat homomorphic encryption and randomization techniques to ensure both efficiency and privacy preservation in positioning. Furthermore, to mitigate the negative effects arising from non-line-of-sight (NLoS) ACs, PPRP achieves privacy-preserving residual-based NLoS analysis. To this end, we additionally propose a series of secure two-party sub-protocols to support various non-linear functions, including comparison, division, square root computation, oblivious shuffle and sorting. These sub-protocols serve as fundamental modules that can be effectively combined to perform sophisticated operations of NLoS analysis in a privacy-preserving manner. A comprehensive simulation-based security analysis demonstrates that PPRP can achieve location privacy preservation. Finally, we develop a proof-of-concept prototype and conduct extensive experiments to show PPRP's high performance in terms of positioning accuracy, computational efficiency, and communication complexity.**

***Index Terms*—Decentralized trust, LoS/NLoS identification, location privacy, multilateration, range-based positioning.**

Manuscript received 8 April 2023; revised 14 November 2023; accepted 6 February 2024. Date of publication 15 February 2024; date of current version 3 September 2024. This work was supported in part by the National Key Research and Development Program of China under Grant 2021ZD0112802, in part by the National Natural Science Foundation of China under Grant 62072215, in part by the Key-Area Research and Development Program of Guangdong Province under Grant 2020B0101090004 and Grant 2020B0101360001, and in part by the Natural Sciences and Engineering Research Council (NSERC) of Canada. Recommended for acceptance by X. Tian. *(Corresponding author: Dongxiao Liu.)*

Cheng Huang is with the School of Computer Science, Fudan University, Shanghai 200438, China.

Dongxiao Liu is with the School of Computer Science and Engineering, University of Electronic Science and Technology of China, Chengdu, Sichuan 611731, China (e-mail: dongxiao.liu@uestc.edu.cn).

Anjia Yang is with the College of Cyber Security, Jinan University, Guangzhou, Guangdong 510632, China, and also with Pazhou Lab, Guangdong 510632, China.

Rongxing Lu is with the Faculty of Computer Science, University of New Brunswick, Fredericton, NB E3B 5A3, Canada.

Xuemin Shen is with the Department of Electrical and Computer Engineering, University of Waterloo, ON N2L 3G1, Canada.

Digital Object Identifier 10.1109/TMC.2024.3366340

I. Introduction

LOCATION-BASED services (LBSs), such as navigation and Points of Interest (POI) recommendations, have become indispensable services in our daily lives, providing tailored and high-quality services based on mobile users' locations [1], [2], [3]. The fundamental prerequisite for LBSs is that users can obtain and upload real-time and precise locations using positioning systems. One of the most famous positioning systems is the global navigation satellite system (GNSS), which allows users to locate themselves by relying on signals transmitted from satellites orbiting Earth. Although GNSS, such as GPS, is highly effective, it may not always be available, particularly in urban areas [4], [5]. Consequently, ubiquitous mobile networks have become an alternative choice, where users can connect to nearby mobile reference points, also known as anchors, and analyze location-dependent parameters between them to achieve self-positioning [6]. Many positioning techniques can work in mobile networks, utilizing various parameters such as time of arrival (ToA), receive signal strength (RSS), and time difference of arrival (TDoA). Among them, range-based positioning based on multilateration, is one of the widely adopted methods [7].

Range-based positioning methods typically require at least three anchors (ACs) and user equipment (UE, referred to as the requested user) to participate in the positioning process. ACs and UE collaboratively utilize measurement data, including range or range-difference measurements between AC and UE, and location information of ACs, to estimate the UE's current location. During this process, location privacy leakages may occur, leading to serious security issues and privacy concerns [8], [9]. On one hand, since ACs may not be limited to fixed base stations and could be moving vehicles or unmanned aerial vehicles, they may not be willing to publicly share their locations and may prefer a privacy-preserving setting [10], [11], [12], [13]. On the other hand, the eventual positioning results may be passively leaked because the computations of location estimation are always conducted by third-party network servers instead of being fully controlled by the requested user. Therefore, many state-of-the-art schemes that aim to achieve privacy-preserving range-based positioning have been proposed recently [14], [15], [16], [17], [18], [19].

These works generally adhere to two positioning frameworks and correspondingly rely on different secure and privacy-preserving techniques. In a pervasive computing framework where privacy-preserving computations are performed by UE and ACs, traditional secure multi-party computation techniques such as garbled circuits can be adopted to achieve secure positioning [19], although they may come at the cost of low efficiency. In addition, there exist a popular lightweight technique known as privacy-preserving summation [14], [17], [20].

1536-1233 © 2024 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission.
See https://www.ieee.org/publications/rights/index.html for more information.

Authorized licensed use limited to: University of Waterloo. Downloaded on February 17,2026 at 23:06:10 UTC from IEEE Xplore. Restrictions apply.

The privacy-preserving summation technique allows multiple ACs and UE to add self-generated noises (randomness) to range/range-difference measurement data. The added noises can then be cancelled out with each other after aggregating the noised measurement data in a particular format. In this way, the privacy is protected as individual's measurement data are hidden by noises before aggregation and also concealed in an aggregated form. With the aggregated measurement data, the estimated location can be computed and obtained by UE. Different from the pervasive framework, another positioning framework needs the deployment of a centralized location management function (LMF) server as an aggregator. The aggregator is responsible for securely collecting UE and ACs' measurement data and performing the positioning. To preserve location privacy, fully/partially homomorphic encryption techniques or secure hardware techniques can be applied, enabling ACs and UE to encrypt measurement data as ciphertexts, and all computations executed at the centralized aggregator are transformed to ciphertext operations or secure enclave operations [18], [21], [22].

Different from the above, we propose a decentralized trust-based positioning framework that divides the trust on a centralized location management function (LMF) server into two independent LMF servers, e.g., two edge servers [23]. Unlike existing works under the pervasive framework, UE and ACs are only responsible for reporting measurement data to LMF servers and can go offline afterwards. In other words, under our proposed decentralized trust-based positioning framework, UE and ACs do not need to participate in the location estimation process. This design allows for the use of lightweight additive secret sharing techniques to upload measurement data in a privacy-preserving manner, significantly reducing computational and communication burdens. Furthermore, all computations of location estimation are outsourced to LMF servers, making the framework suitable for IoT scenarios with resource-constrained devices. Since two LMF servers are deployed and can interact with each other for achieving privacy-preserving computations, heavy ciphertext computations introduced by homomorphic multiplications can also be evaded to some extent, which makes location estimation more computation-efficient than existing works that employ fully homomorphic encryption under the centralized framework [22].

With such a decentralized framework, a remaining issue is how to construct a compatible and efficient privacy-preserving range-based positioning scheme. To tackle the issue, we begin by disassembling the procedures of location estimation into matrix computations, and then achieve privacy-preserving location estimation using an extension and hybrid combination of two secure two-party sub-protocols for computing matrix multiplication and matrix inverse. These two sub-protocols are carefully tailored based on somewhat homomorphic encryption and randomization techniques, and it can be noted that, by setting proper parameters, the somewhat homomorphic encryption can be a better choice than the common Paillier homomorphic encryption and can provide higher computational efficiency in our decentralized framework. In addition, we address the issue of how to perform privacy-preserving non-line-of-sight (NLoS) analysis. The issue is challenging, considering that LoS/NLoS identification algorithms are complex and involve different kinds of sophisticated no-linear computations such as comparison, division, square root computation, oblivious shuffle and sorting. To overcome the challenge, we further design several secure two-party computation sub-protocols by elaborately converting plaintext algorithms into ciphertext algorithms. The proposed sub-protocols are then utilized as basic secure modules to realize privacy-preserving LoS/NLoS identification based on residual analysis [24], [25]. In summary, the contributions of the paper are three-folds:

- First, a set of secure and efficient two-party computation sub-protocols are proposed as basic modules to achieve essential functions such as comparison, square computation, matrix multiplication, integer mapping, matrix inverse, absolute value difference, division, square root computation, oblivious shuffle, and quicksort. These sub-protocols are distinct from other secure two-party sub-protocols that rely on beaver triples. Instead, they are constructed entirely using somewhat homomorphic encryption, which eliminates the need for time-consuming offline preprocessing. These sub-protocols may be applied to other schemes with independent interest.
- Second, we introduce a decentralized trust-based framework into range-based positioning and propose a privacy-preserving range-based positioning scheme under the framework, named PPRP. PPRP differs from existing works in that it utilizes lightweight techniques that are suitable for devices with limited resources. Additionally, PPRP supports privacy-preserving LoS/NLoS identification based on residual analysis, which is a more efficient and effective approach compared to other methods.
- Third, we define a simulation-based location privacy model for range-based positioning, and prove that PPRP achieves location privacy preservation under a semi-honest adversarial model. Also, we develop a proof-of-concept prototype using Python to demonstrate that PPRP achieves high performance in terms of positioning accuracy, computational efficiency, and communication complexity.

The remainder of this paper is organized as follows. In Section II, we introduce the system model, define the adversarial and security models, and identify the design goals. Then, we present preliminaries in Section III and propose a series of secure two-party sub-protocols as basic modules in Section IV. Next, we give the detailed constructions of the proposed privacy-preserving range-based positioning scheme (PPRP) in Section V. Subsequently, security analysis and performance evaluation are presented in Sections VI and VII, respectively. Finally, Section VIII reviews some related works and Section IX draws the conclusion.

II. Models and Design Goals

A. Range-Based Positioning Model

Range-based positioning is a widely adopted positioning method in mobile networks, in which $N \geq 3$ nearby anchors (ACs) can collaborate with each other and user equipment (UE) to help position UE's locations with range measurements or range-difference measurements. For various positioning scenarios, ACs may be cellular base stations [26] or other mobile probes such as unmanned aerial vehicles [27].

A high-level procedure of range-based positioning is as follows: 1) UE first broadcasts a signal which can be captured by nearby N ACs; 2) among N ACs ($\mathrm{AC}_1, \mathrm{AC}_2, \ldots, \mathrm{AC}_N$), one AC is chosen as the home AC (AC_1), and other neighboring ACs are the non-home ACs; 3) UE and N ACs can perform range/range-difference measurements according to location independent parameters such as time of arrival (ToA), time difference of arrival (TDoA), and received signal strength (RSS); and

Authorized licensed use limited to: University of Waterloo. Downloaded on February 17,2026 at 23:06:10 UTC from IEEE Xplore. Restrictions apply.

4) with the inputs of measurement data, least-squares methods can be applied to estimate the location of UE [28].

Without loss of generality, we use (x_i, y_i) to denote the location of AC_i ($i = 1$ to N) and use (x_0, y_0) to denote the location of UE, and there exist two conventional least-squares (LS) location estimation methods for range measurements and range-difference measurements: *SR-LS* and *SRD-LS* [28].

SR-LS Method: The range (distance) between AC_i ($i = 1$ to N) and UE is measured, and is denoted by $\tilde{d}_i = d_i + e_i$, where e_i is the measurement error and $d_i = \sqrt{(x_i - x_0)^2 + (y_i - y_0)^2}$ is the exact euclidean distance between AC_i and UE. With the measured distances and exact locations of ACs, an SR-LS method can be applied to estimate UE's location $\mathbf{z} = (x_0, y_0)$:

$$\mathbf{z} = (\mathbf{A}^T\mathbf{A})^{-1}\mathbf{A}^T\mathbf{b}, \tag{1}$$

where $\mathbf{b} = (x_2^2 + y_2^2 - x_1^2 - y_1^2 - \tilde{d}_2^2 + \tilde{d}_1^2, x_3^2 + y_3^2 - x_1^2 - y_1^2 - \tilde{d}^2 + \tilde{d}_1^2, \ldots, x_N^2 + y_N^2 - x_1^2 - y_1^2 - \tilde{d}_N^2 + \tilde{d}_1^2)^T$, and

$$\mathbf{A} = \begin{bmatrix} 2(x_2 - x_1) & 2(y_2 - y_1) \\ 2(x_3 - x_1) & 2(y_3 - y_1) \\ \vdots & \vdots \\ 2(x_N - x_1) & 2(y_N - y_1) \end{bmatrix}. \tag{2}$$

SRD-LS Method: The range-difference measurement between AC_i ($i = 2$ to N) and AC_1 is represented by $\tilde{d}_{i1} = |d_i - d_1| + e_{i1}$, where e_{i1} is the measurement error. Similarly, with the measured differences and exact locations of ACs, an SRC-LS method can be applied to estimate UE's location $\mathbf{z} = (x_0, y_0, d_1)$:

$$\mathbf{z} = \frac{1}{2}(\mathbf{A}^T\mathbf{A})^{-1}\mathbf{A}^T\mathbf{b}, \tag{3}$$

where $\mathbf{b} = (\tilde{d}_{21}^2 - x_2^2 - y_2^2 + x_1^2 + y_1^2, \tilde{d}_{31}^2 - x_3^2 - y_3^2 + x_1^2 + y_1^2, \ldots, \tilde{d}_{N1}^2 - x_N^2 - y_N^2 + x_1^2 + y_1^2)^T$, and

$$\mathbf{A} = \begin{bmatrix} x_1 - x_2 & y_1 - y_2 & \tilde{d}_{21} \\ x_1 - x_3 & y_1 - y_3 & \tilde{d}_{31} \\ \vdots & \vdots & \vdots \\ x_1 - x_N & y_1 - y_N & \tilde{d}_{N1} \end{bmatrix}. \tag{4}$$

Based on the latest 3GPP standard [29], there are two general system models for range-based positioning : 1) (Downlink model denoted by DL) UE is responsible for range/range-difference measurements and 2) (Uplink model denoted by UL) ACs perform range/range-difference measurements. For both models, there exists a network entity named Location Management Function (LMF) that is responsible for configuring and managing positioning services. In reality, LMF can be edge servers deployed in mobile networks [30] (our system model requires two servers denoted by S_0 and S_1). They interact with the serving ACs and UE to collect range/range-difference measurement data, and are responsible for calculating the estimated location of UE. Accordingly, we present a generalized system model as shown in Fig. 1 and use Table I to show the range measurement data (RM) and the range-difference (RDM) measurement data reported by ACs and UE ($N = 3$) in DL and UL models.

LoS/NLoS Identification: Range-based positioning methods suffer from non-line-of-sight (NLoS) propagation error, one of the dominant factors that affects e_i or e_{i1}. Hence, it is desirable to identify LoS ACs and only rely on LoS ACs to perform location

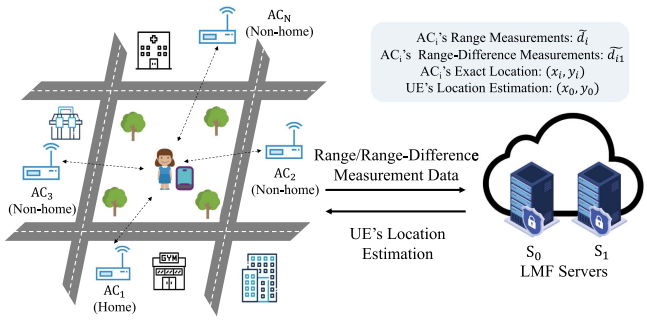


Fig. 1. Generalized system model for range-based positioning.

TABLE I
UE AND ACS' RANGE/RANGE-DIFFERENCE MEASUREMENT DATA IN DL AND UL MODELS ($N = 3$)

Entities	UE	AC_1	AC_2	AC_3
DL (RM)	$\tilde{d}_1, \tilde{d}_2, \tilde{d}_3$	(x_1, y_1)	(x_2, y_2)	(x_3, y_3)
DL (RDM)	$\tilde{d}_{21}, \tilde{d}_{31}$	(x_1, y_1)	(x_2, y_2)	(x_3, y_3)
UL (RM)	N/A	$(x_1, y_1), \tilde{d}_1$	$(x_2, y_2), \tilde{d}_2$	$(x_3, y_3), \tilde{d}_3$
UL (RDM)	N/A	(x_1, y_1)	$(x_2, y_2), \tilde{d}_{21}$	$(x_3, y_3), \tilde{d}_{31}$

estimation. One common LoS/NLoS identification algorithm is designed based on residuals [24], [25], and is effective even without any knowledge of the NLoS error under the assumption that a small subset of the total available ACs are NLoS ACs. Specifically, a residual-based identification algorithm has the following steps:

1) Use range/range-difference measurement data from all N ACs to calculate the estimated location of UE based on (1) or (3) as a reference location, $(\hat{x}_0, \hat{y}_0)$.
2) Traverse each subset of N ACs: each subset $\mathbb{S}_j$ includes at least 3 (or 4) ACs, and one of them is the home AC, and the total number of subsets is $M = \binom{N-1}{K-1}$, $3 \le K < N$ (or $4 \le K < N$), i.e., $j = 1, 2, \ldots, M$.
3) For each subset $\mathbb{S}_{j \in [1,M]}$, calculate the residual, RS_j, using the reference location. For *SR-LS*, the residual is
$$\text{RS}_j = \sum_{\text{AC}_i \in \mathbb{S}_j} (\tilde{d}_i - \hat{d}_i)^2, \tag{5}$$
where $\hat{d}_i = \sqrt{(x_i - \hat{x}_0)^2 + (y_i - \hat{y}_0)^2}$.
For *SRD-LS*, the residual is
$$\text{RS}_j = \sum_{\text{AC}_i \in \mathbb{S}_j} (\tilde{d}_{i1} - \hat{d}_{i1})^2, \tag{6}$$
where $\hat{d}_{i1} = |\hat{d}_i - \hat{d}_1|$.
4) Sum the residuals of the subsets that AC_i belong to for $i = 1$ to N, and the summation is denoted by W_i.
5) Rank N ACs according to the summations, and pick top-$\bar{N}$ ACs as possible LoS ACs with the smallest summation of residuals.
6) Perform location estimation using top-$\bar{N}$ ACs' range/range-difference measurement data.

B. Adversarial Model & Security Model

Adversarial Model: This work follows an honest-but-curious (a.k.a. semi-honest) adversarial model that is adopted in most of

Authorized licensed use limited to: University of Waterloo. Downloaded on February 17,2026 at 23:06:10 UTC from IEEE Xplore. Restrictions apply.

the state-of-the-art privacy-preserving positioning schemes [14], [17], [19], [21], [31]. The honest-but-curious model means that all ACs and UE honestly follow the positioning protocols, but part of these entities may be passive adversaries who are curious about other honest entities' location information. More specifically, there exist two types of attacks: 1) (Attack-I) UE, a set of ACs, and one LMF server are controlled by adversaries who aim to infer other honest ACs' locations; and 2) (Attack-II) a set of ACs and one LMF server are controlled by adversaries who aim to extract UE's location. Since our work focuses on privacy preservation, other attacks are beyond the scope of this work. Meanwhile, we assume that at least one LMF server (S_0 or S_1) is not compromised by the adversaries, since they are deployed independently. The assumption is reasonable and widely acknowledged in the security community [32]. Also, we assume that there exists a secure and authenticated communication channel among UE, ACs, and LMF servers.

Syntax of Privacy-Preserving Range-Based Positioning. A privacy-preserving range-based positioning scheme, Π, mainly consists of one algorithm, `Initialize`, and two protocols, `Filter` and `Estimate`:

- $\texttt{Initialize}(u_i) \rightarrow (u_i'; u_i'')$*:* The measurement initialization algorithm is executed by UE and N ACs. The inputs of the algorithm are UE's plaintext measurement data denoted by u_0 (resp. AC_i's plaintext measurement data denoted by u_i, $i = 1$ to N). The outputs, $(u_i'; u_i'')$, are additively secret-shared measurement data, where u_i' is reported to S_0 and u_i'' is reported to S_1;
- $\texttt{Filter}((u_i')_{i=0}^{N}; (u_i'')_{i=0}^{N}; \bar{N}) \rightarrow ((v_i')_{i=1}^{\bar{N}}; (v_i'')_{i=1}^{\bar{N}})$*:* The privacy-preserving LoS/NLoS identification protocol is executed between S_0 and S_1 for locating top-$\bar{N}$ (out of N) possible LoS ACs based on residuals. The inputs of LMF servers include secret-shared measurement data received from UE and ACs, $(u_i')_{i=0}^{N}$ and $(u_i')_{i=0}^{N}$. The outputs are $\bar{N}$ secret-shared ACs' indexes, $(v_i')_{i=1}^{\bar{N}}$ and $(v_i'')_{i=1}^{\bar{N}}$, stored in S_0 and S_1, respectively. Combining v_i' and v_i'' for $i = 1$ to $\bar{N}$ will obtain the indexes of top-$\bar{N}$ possible LoS ACs, satisfying $v_i' + v_i'' \in [1, N]$;
- $\texttt{Estimate}((u_i')_{i=0}^{N}, (v_i')_{i=1}^{\bar{N}}; (u_i'')_{i=0}^{N}, (v_i'')_{i=1}^{\bar{N}}; \mathsf{sign}) \rightarrow (x_0, y_0)$*:* The privacy-preserving location estimation protocol is performed between LMF servers (S_0, S_1) and UE. At the end of the protocol, UE obtains the estimated location, (x_0, y_0). UE uses $\mathsf{sign} \in \{0, 1\}$ to indicate whether LoS/NLoS identification is used ($\mathsf{sign} = 0$ means not using LoS/NLoS identification) when performing location estimation.

Security Model: The privacy leakages are captured by a family of stateful leakage functions $\mathcal{L} = (\mathcal{L}^{\text{init}}, \mathcal{L}^{\text{filter}}, \mathcal{L}^{\text{est}})$ that describe what information is leaked to a polynomial-probabilistic-time (PPT) adversary $\mathcal{A}$ during the executions of `Initialize`, `Filter`, and `Estimate`. Based on the leakage functions, the security model is formulated using a simulation where a real world $\mathsf{REAL}_{\mathcal{A}}^{\Pi}(\lambda)$ and an ideal world $\mathsf{IDEAL}_{\mathcal{A},\mathcal{S},\mathcal{L}}^{\Pi}(\lambda)$ exist (λ is a security parameter):

- $\mathsf{REAL}_{\mathcal{A}}^{\Pi}(\lambda)$*:* For Attack-II, honest UE, honest ACs and corrupted ACs (controlled by $\mathcal{A}$) run $\texttt{Initialize}((u_i)_{i=0}^{N})$ to report $(u_i')_{i=0}^{N}$ and $(u_i'')_{i=0}^{N}$ to S_b and S_{1-b} (controlled by $\mathcal{A}$), where $b \in \{0, 1\}$. Then, honest S_b and $\mathcal{A}$ perform `Filter` to find top-$\bar{N}$ possible LoS ACs. Finally, honest UE, honest S_b, and $\mathcal{A}$ perform `Estimate` to achieve location estimation. All transcripts generated by running `Initialize`, `Filter` and `Estimate` are revealed to $\mathcal{A}$, which are denoted by $\mathsf{View}_{\mathcal{A}}^{\mathsf{REAL}}$. For Attack-I, $\mathcal{A}$ additionally controls UE, and the procedures are similar.
- $\mathsf{IDEAL}_{\mathcal{A},\mathcal{S},\mathcal{L}}^{\Pi}(\lambda)$*:* For Attack-II, a simulator $\mathcal{S}$ (that behaves as honest UE, honest ACs, and honest S_b) works with $\mathcal{A}$ (that controls corrupted ACs and S_{1-b}) to simulate `Initialize`, `Filter` and `Estimate` by exploiting $\mathcal{L}$. All transcripts generated when simulating `Initialize`, `Filter` and `Estimate` are revealed to $\mathcal{A}$, which are denoted by $\mathsf{View}_{\mathcal{A},\mathcal{S},\mathcal{L}}^{\mathsf{IDEAL}}$. For Attack-I, $\mathcal{S}$ does not need to simulate honest UE, since UE is controlled by $\mathcal{A}$.

If $\mathcal{A}$ cannot distinguish two worlds, it is obvious that there is no information leaked except $\mathcal{L}$, and accordingly, we give a privacy definition in the following:

Definition 1 (Location Privacy): A range-based positioning scheme, Π, is $\mathcal{L}$-semantically-secure if, for a PPT adversary, $\mathcal{A}$, there exists a PPT algorithm $\mathcal{S}$ such that

$$\mathsf{View}_{\mathcal{A}}^{\mathsf{REAL}} \overset{c}{\approx} \mathsf{View}_{\mathcal{A},\mathcal{S},\mathcal{L}}^{\mathsf{IDEAL}}.$$

C. Design Goals

Our goal is to design a privacy-preserving range-based positioning scheme which can also guarantee practical utility requirements:

- *Positioning Error:* Without privacy preservation, positioning errors are mainly due to measurement errors and estimation errors. The scheme should not introduce much positioning accuracy loss, except the original positioning loss in plaintext-based schemes.
- *UE/ACs Computational Efficiency:* Computations at UE/AC side should be lightweight in the scheme since UE/ACs could be an energy-constraint sensor or other IoT devices.
- *End-to-End Execution Delay:* The end-to-end execution delay to obtain the eventual UE's location estimation should be low, since positioning is a real-time service in most of the applications.
- *Communication Complexity:* The communication complexity is defined as end-to-end communication rounds between UE/ACs and LMF servers. The communication complexity should not be high.

III. Preliminaries

In this section, we review two cryptographic primitives: two-party additive secret sharing (ASS) and somewhat homomorphic encryption (SHE).

A. Two-Party Additive Secret Sharing

Additive secret sharing (ASS) [33] is widely used in secure two-party computation as a basic module, and mainly consists of three procedures:

- *Secret Distribution:* A secret $s \in [-2^{l-1}, 2^{l-1}]$ can be randomly split into two secret shares, $s' \in \mathbb{Z}_{2^l}$ and $s'' \in \mathbb{Z}_{2^l}$, satisfying $s' + s'' = s \bmod 2^l$. In a two-party scenario, s' and s'' are distributed to two parties, respectively.
- *Secret Reconstruction:* Two parties can contribute their secret shares, s' and s'', to recover $s = s' + s'' \bmod 2^l$ if $s \leq 2^{l-1}$ and $s = s' + s'' - 2^l$ if $s > 2^{l-1}$.
- *Homomorphic Addition:* Suppose that secrets s_1 and s_2 are distributed to two parties, each party can compute the

Authorized licensed use limited to: University of Waterloo. Downloaded on February 17,2026 at 23:06:10 UTC from IEEE Xplore. Restrictions apply.

secret shares of $s_1 + s_2$ by performing $s_1' + s_2' \bmod 2^l$ (resp. $s_1'' + s_2'' \bmod 2^l$) without interacting with each other.

B. Somewhat Homomorphic Encryption

Somewhat Homomorphic Encryption (SHE), a.k.a., leveled homomorphic encryption, mainly involves three algorithms: key generation, encryption, and decryption:

- (*Key Generation*) $\texttt{SHE.KeyGen}(\lambda) \rightarrow (\mathsf{sk}, \mathsf{pp})$*:* By inputting security parameters $\lambda = (k_0, k_1, k_2, k_3)$, the algorithm generates a symmetric key $\mathsf{sk} = (p, q, L)$, where p is a safe prime with bit length k_0, q is the product of multiple safe primes with bit length k_0 ($|q| = k_3 = \sum k_0$), and L is a random number with bit length k_1. The algorithm also publishes public parameters $\mathsf{pp} = (k_0, k_1, k_2, k_3, \mathcal{N})$, where $\mathcal{N} = pq$.
- (*Encryption*) $\texttt{SHE.Enc}(m, \mathsf{sk}) \rightarrow c$*:* Given a message m in a plaintext space $\mathcal{M} \in [-2^{k_1-5}, 2^{k_1-5}]$, the algorithm chooses two random numbers (r, r^*) where $r \in \{0,1\}^{k_2}$ and $r^* \in \mathbb{Z}_{\mathcal{N}}$, and generates m's ciphertext as $\mathsf{ct} = [\![m]\!] = m + rL + r^* p \bmod \mathcal{N}$.
- (*Decryption*) $\texttt{SHE.Dec}(\mathsf{ct}, \mathsf{sk}) \rightarrow m$*:* Given a ciphertext ct, the algorithm calculates $m' = (\mathsf{ct} \bmod p) \bmod L$. If $m' < \frac{L}{2}$, the algorithm outputs $m = m'$; otherwise, outputs $m = m' - L$.

SHE supports four homomorphic operations: 1) (Add-I) $[\![m_1]\!] + [\![m_2]\!] = [\![m_1 + m_2]\!]$; 2) (Add-II) $m_1 + [\![m_2]\!] = [\![m_1 + m_2]\!]$; 3) (Mul-I) $[\![m_1]\!] \cdot [\![m_2]\!] = [\![m_1 m_2]\!]$; and 4) (Mul-II) $m_1 \cdot [\![m_2]\!] = [\![m_1 m_2]\!] (m_1 > 0)$. For the operation Mul-I, the maximum depth is $\lfloor \frac{k_0}{k_1+k_2} \rfloor - 1$.

(*Public-key Transformation*) SHE can be transformed to a public-key scheme where a public key $\mathsf{pk} = (\mathsf{pp}, [\![0]\!], [\![0]\!]^*)$ is published. With the public key, the encryption procedure becomes $\mathsf{ct} = [\![m]\!] = m + r_1 [\![0]\!] + r_2 [\![0]\!]^*$ where $r_1 \in \{0,1\}^{k_2}$ and $r_2 \in \{0,1\}^{k_2}$ are two random numbers. SHE additionally supports negate operations, i.e., a symmetric-key ciphertext, $\mathsf{ct} = [\![m]\!]$, can be transformed to a public-key ciphertext of $-m$, $[\![-m]\!] = -\mathsf{ct} + 4 \cdot [\![0]\!]$ ($+4 \cdot [\![0]\!]$ is to ensure $\alpha > 0$ and decryption correctness). Using a similar idea, we can achieve $m_1 \cdot [\![m_2]\!]_0 = m_1 \cdot [\![m_2]\!] + 2^{k_0-k_1-k_2-5} \cdot [\![0]\!] = [\![m_1 m_2]\!]$ with $m_1 < 0$.

The security of SHE relies on the (L, p)-based decision problem and the (L, p)-based decision assumption [34]:

Definition 2 ((L, p)-based Decision Problem): Suppose there exist two sets $\mathbb{S}$ and $\bar{\mathbb{S}}$:

$$\begin{cases} \mathbb{S} : \{\alpha L + \beta p \bmod \mathcal{N} | \alpha, \beta \in \mathbb{Z}_{\mathcal{N}}, \alpha L < p\} \\ \bar{\mathbb{S}} = \mathbb{Z}_{\mathcal{N}} \backslash \mathbb{S} : \{\alpha L + \beta p \bmod \mathcal{N} | \alpha, \beta \in \mathbb{Z}_{\mathcal{N}}, \alpha L \geq p\}, \end{cases}$$

given $\lambda = (k_0, k_1, k_2, k_3)$, the (L, p)-based decision problem is the problem of determining whether an element $w \in \mathbb{Z}_{\mathcal{N}}$ belongs to $\mathbb{S}$ or $\bar{\mathbb{S}}$.

The intractability of (L, p)-based decision problem is established on three conditions: 1) k_0 and k_3 should be chosen to guarantee that factoring $\mathcal{N}$ is computationally difficult, e.g., $k_0 \geq 500$ and $k_0 + k_3 \geq 2048$; 2) k_1 and k_2 should be selected to resist brute-force attacks, e.g., $k_1 \geq 80$, $k_2 \geq 80$; and 3) $\tau = k_0 + k_3$ should be set to thwart various lattice-based attacks [35], i.e., $\tau \geq k_0^2$ and $\tau \geq \frac{\lambda}{\log(\lambda)}(k_0 - k_1 - k_2)^2$. Based on the intractability of the (L, p)-based decision problem, the (L, p)-based decision assumption is given:

Definition 3 ((L, p)-based Decision Assumption [34]): The advantage of polynomial-time adversaries in solving the (L, p)-based decision problem is a negligible function, $\texttt{neg}(\lambda)$, given $\lambda = (k_0, k_1, k_2, k_3)$ satisfying the above-mentioned three conditions.

Theorem 1 (IND-CPA Security [34]): SHE is semantically secure against chosen-plaintext attacks (CPA) under the (L, p)-based decision assumption.

Remarks: Fully homomorphic encryption such as CKKS [22], can support more operations than partially homomorphic encryption such as Paillier, at the expense of tolerating more computational delay. Also, due to the calculations of modular exponentiation, Paillier is somehow time-consuming. In this paper, SHE is leveraged since we observe that SHE is more computation-efficient than Paillier, especially combining with ASS. When SHE's parameters are properly chosen, higher computation efficiency can be satisfied under the decentralized trust. More specifically, the bottleneck of SHE is the size of τ. To thwart lattice-based attacks, τ should be larger than $\frac{\lambda}{\log(\lambda)}(k_0 - k_1 - k_2)^2$ (λ is the security parameter). For instance, τ should be at least 4,304,578 to achieve 80-bit security with $k_0 = 500$, $k_1 = 200$, and $k_2 = 100$. The increase of τ not only causes high computational latency but also leads to larger ciphertext expansion and communication overheads. To reduce the size of τ, we set $\tau = k_0^2$ (i.e., $k_3 = k_0^2 - k_0$) and ensure that a symmetric key (p, q, L) is used fewer than t times during range-based positioning, where $t + 1 < \frac{\tau - k_1 - k_2}{k_0 - k_1 - k_2}$. After reaching the limitation, a new symmetric key is adopted. This setting is reasonable under the decentralized framework, in which two parties can self-generate their keys and share public keys with each other. With such a setting, we can reduce τ from 4,304,578 to 250,000 and significantly improve the computational efficiency while thwarting lattice-based attacks. Table II shows the computation efficiency comparison between SHE and Paillier (2048 bits) in terms of encryption, decryption, and ciphertext operations (implemented using Python). Certainly, SHE sacrifices some communication efficiency, i.e., its ciphertext size is larger than Paillier. If applications care more about communication efficiency, SHE can be seamlessly replaced by Paillier as we completely circumvent the homomorphic Mul-I operation in our proposed scheme.

TABLE II
EFFICIENCY COMPARISON BETWEEN SHE AND PAILLIER: UNITS (MS)

Scheme	Enc_{Sym}	Enc_{Pk}	Dec	Add-I	Add-II	Mul-I	Mul-II
Paillier	N/A	6.54	1.78	0.007	0.01	N/A	0.079
SHE^1	0.24	0.146	0.08	0.008	0.005	N/A	0.014
SHE^2	0.537	0.304	0.2	0.019	0.012	37.659	0.031

Note: SHE^1 is with $(k_0 = 500, k_1 = 200, k_2 = 100, t = 1245)$ which does not support homomorphic Mul-I operations; SHE^2 is with $(k_0 = 750, k_1 = 250, k_2 = 100, t = 1402)$ which supports homomorphic Mul-I operations.

IV. BASIC MODULES: SECURE SUB-PROTOCOLS

Before presenting the proposed positioning scheme, we first present a series of secure two-party sub-protocols as basic modules, including a secure comparison (SLT/SGET) protocol, a secure square (SS) protocol, a secure matrix multiplication (SMM) protocol, a secure integer mapping (SMAP) protocol, a secure

Authorized licensed use limited to: University of Waterloo. Downloaded on February 17,2026 at 23:06:10 UTC from IEEE Xplore. Restrictions apply.

TABLE III
SUMMARY OF BASIC MODULES

Module	Function
Secure Comparison (SLT/SGET)	The protocol enables two servers to securely compare secret-shared numbers and obtain secret-shared comparison results.
Secure Square (SS)	The protocol allows two servers to securely calculate the squares of secret-shared numbers, with the squared results kept secret-shared.
Secure Matrix Multiplication (SMM)	The protocol enables matrix multiplication on secret-shared matrices, and the multiplication results are secret shared.
Secure Integer Mapping (SMAP)	The protocol allows two servers to achieve secure transformation of ciphertexts from the modulo domain to the integer domain.
Secure Matrix Inverse (SMI)	The protocol allows two servers to securely calculate the inverse of secret-shared matrices, while keeping the inverted matrix secret-shared.
Secure Integer Division (SIDIV)	The protocol enables the computation of floor-rounded division on secret-shared numbers through two-server interaction, with the division results kept secret-shared.
Secure Absolute Value Difference (SAVD)	The protocol allows two servers to securely compute the absolute difference of secret-shared numbers, and secretly shares the results.
Secure Integer Square Root (SISR)	The protocol allows two servers to securely calculate the secret-shared square root of secret-shared numbers.
Secure Oblivious Shuffle (SOS)	The protocol enables two servers to perform a shuffle of secret-shared key-value pairs without revealing the permutations or the pairs, with the results remaining secret-shared.
Secure Quicksort (SQ)	The protocol enables securely sorting of secret-shared key-value pairs through two-server interaction, without exposing keys or values, and the sorted results are secret shared.

matrix inverse (SMI) protocol, a secure integer division (SIDIV) protocol, a secure absolute value difference (SAVD) protocol, a secure integer square root (SISR) protocol, a secure oblivious shuffle (SOS) protocol, and a secure quicksort (SQ) protocol. For the reader's convenience, we summarize the functions of the sub-protocols in Table III.

The proposed secure sub-protocols have a common setting where two servers (S_0 and S_1) exist, and each server maintains a public/private key pair generated using $\mathtt{SHE.KeyGen}(\lambda)$, i.e., S_0 has $(\mathsf{pk}_0, \mathsf{sk}_0)$, S_1 possesses $(\mathsf{pk}_1, \mathsf{sk}_1)$, and they know each other's public key. Their keys can be refreshed after using at most $t = \lfloor \frac{\tau - k_1 - k_2}{k_0 - k_1 - k_2} - 1 \rfloor$ times to resist lattice-based attacks. We set ciphertexts encrypted by S_0 as $[\![\cdot]\!]_0$ and ciphertexts encrypted by S_1 as $[\![\cdot]\!]_1$.

Comparison With Existing Methods: The proposed protocols offer three key benefits over conventional secure multi-party computation (SMC) methods and fully homomorphic encryption (FHE): 1) *No Preprocessing:* Our protocols eliminate the need for preprocessing — unlike Beaver's triples-based SMC — and enable immediate execution between any two servers without prior interaction or extensive preparation; 2) *Flexibility:* Compared to SMC based on garbled circuits and oblivious transfer, which is designed for specific tasks with limited reusability, our protocols offer a modular design and allow for arbitrary composition to ensure flexibility; and 3) *Efficiency:* Our protocols are more computationally efficient than FHE, significantly reducing computational costs at the expense of some communication overheads.

Protocol 1: Secure Comparison (SLT/SGET).

Input: S_0:$(\alpha', \beta', \mathsf{sk}_0)$; S_1:$(\alpha'', \beta'', \mathsf{pk}_0)$
Output: S_0:ϕ'; S_1:ϕ''
1: S_0 sets $w' = a_0 - b_0$ for secure less-than (SLT) comparison or $w' = b_0 - a_0$ for secure greater-than-equal-to (SGET) comparison, and shares $[\![w']\!]_0$ with S_1
2: S_1 sets $w'' = a_1 - b_1$ (SLT) or $w'' = b_1 - a_1$ (SGET), flips a coin $b \in \{0, 1\}$, sets $\phi'' = 1$ if $b = 1$ or $\phi'' = 0$ if $b = 0$, chooses two random numbers $r_1, r_2 \in \mathbb{Z}_{2^l}$ $(r_1 > r_2)$, computes

$$\hat{\mathsf{ct}} = \begin{cases} r_1 \cdot ([\![w']\!]_0 + w'') + r_2 & \text{if } b = 1; \\ r_1 \cdot (-[\![w']\!]_0 + 4 \cdot [\![0]\!]_0 + w'') - r_2 & \text{if } b = 0, \end{cases} \tag{7}$$

and returns $\hat{\mathsf{ct}}$ to S_0
3: S_0 decrypts $\hat{\mathsf{ct}}$: if $\mathtt{Dec}(\hat{\mathsf{ct}}) < 0$, sets $\phi' = 0$; otherwise $\phi' = 1$

Protocol 2: Secure Square (SS).

Input: S_0:(α', sk_0); S_1:$(\alpha'', \mathsf{pk}_0)$
Output: S_0:ω'; S_1:ω''
1: S_0 shares $[\![\alpha']\!]_0$ with S_1
2: S_1 chooses $r \in \mathbb{Z}_{2^l}$, sets $\omega'' = (\alpha'')^2 - r \bmod 2^l$, and returns $2\alpha'' \cdot [\![\alpha']\!]_0 + r$ to S_0
3: S_0 sets $\omega' = (\alpha')^2 + \mathtt{Dec}(2\alpha'' \cdot [\![\alpha']\!]_0 + r) \bmod 2^l$

A. Secure Comparison

The secure comparison (SLT/SGET) protocol is a generalized version of the protocol originally proposed in Section IV-B.1 of [36], and is modified slightly to enable less-than/greater-than-equal-to comparison without homomorphic Mul-I operations. Suppose that S_0 has α' and β' and S_1 has α'' and β'', the protocol allows two parties to determine whether $\alpha < \beta$ (resp. $\alpha \geq \beta$), and use one bit $\phi \in \{0, 1\}$ to indicate the comparison result. If $\alpha < \beta$ (resp. $\alpha \geq \beta$), $\phi = 1 = \phi' \oplus \phi''$; otherwise; $\phi = 0 = \phi \oplus \phi''$. At the end of the protocol, S_0 gets ϕ' and S_1 obtains ϕ''. The details of the protocol are given in Protocol 1. We omit the correctness analysis of the protocol, which can be found in Section IV-B.1 of [36].

B. Secure Square

The secure square (SS) protocol is a generalized version of the protocol originally proposed in Section IV-B.3 of [36]. Suppose that S_0 has α' and S_1 has α'', the protocol allows two parties to calculate $\alpha^2 = (\alpha' + \alpha'')^2 = \omega' + \omega''$. At the end of the protocol, S_0 gets ω' and S_1 obtains ω'', without leaking α to any party. The details of the protocol are present in Protocol 2, and its correctness is obvious.

C. Secure Matrix Multiplication

The detailed secure matrix multiplication protocol is presented in Protocol 3. The protocol enables two parties securely calculating the multiplication of an additively secret-shared

Authorized licensed use limited to: University of Waterloo. Downloaded on February 17,2026 at 23:06:10 UTC from IEEE Xplore. Restrictions apply.

Protocol 3: Secure Matrix Multiplication (SMM).

Input: S_0:$(\mathbf{A}', \mathbf{B}', \mathsf{sk}_0)$; S_1:$(\mathbf{A}'', \mathbf{B}'', \mathsf{pk}_0)$
Output: S_0: $\mathbf{C}'$; S_1:$\mathbf{C}''$
1: S_0 encrypts $\mathbf{A}'$ and $\mathbf{B}'$: $[\![\mathbf{A}']\!]_0 = ([\![a'_{ij}]\!]_0)_{i=1,j=1}^{n_1,n_2}$, $[\![\mathbf{B}']\!]_0 = ([\![b'_{ij}]\!]_0)_{i=1,j=1}^{n_2,n_1}$, and shares $[\![\mathbf{A}']\!]_0$ and $[\![\mathbf{B}']\!]_0$ with S_1
2: S_1 performs homomorphic additions and multiplications: $[\![\mathbf{W}]\!]_0 = [\![\mathbf{A}']\!]_0 \times \mathbf{B}'' + \mathbf{A}'' \times [\![\mathbf{B}']\!]_0$; generates an $n_1 \times n_1$ random matrix $\mathbf{R} = (r_{ij})_{i=1,j=1}^{n_1,n_1}$, where $r_{ij} \in \mathbb{Z}_{2^l}$ is a random number; and sets $\mathbf{C}'' = \mathbf{A}'' \times \mathbf{B}'' - \mathbf{R} \bmod 2^l$ and shares $[\![\mathbf{W}]\!]_0 + \mathbf{R}$ with S_0
3: S_0 decrypts $[\![\mathbf{W}]\!]_0 + \mathbf{R}$ to obtain $\mathbf{W} - \mathbf{R}$ and $\mathbf{C}' = \mathbf{W} + \mathbf{R} + \mathbf{A}' \times \mathbf{B}' \bmod 2^l$

$n_1 \times n_2$ matrix, $\mathbf{A} = (a_{ij})_{i=1,j=1}^{n_1,n_2}$ ($\mathbf{A} = \mathbf{A}' + \mathbf{A}'' \bmod 2^l$) and an additively secret-shared $n_2 \times n_1$ matrix, $\mathbf{B} = (b_{ij})_{i=1,j=1}^{n_2,n_1}$ ($\mathbf{B} = \mathbf{B}' + \mathbf{B}'' \bmod 2^l$). Finally, S_0 gets $\mathbf{C}'$ and S_1 obtains $\mathbf{C}''$, satisfying $\mathbf{C} = \mathbf{A} \times \mathbf{B} = \mathbf{C}' + \mathbf{C}'' \bmod 2^l$, where $\mathbf{C}$ is an $n_1 \times n_1$ matrix. The idea behind the protocol is as follows: for two secret-shared values $a_1 = a'_1 + a''_1$ and $b_1 = b'_1 + b''_1$, we have their product, $w = a_1 * b_1 = (a'_1 + a''_1)(b'_1 + b''_1) = a'_1 b'_1 + a''_1 b'_1 + a'_1 b''_1 + a''_1 b''_1$; to construct the secret shares of the product, S_0 and S_1 independently calculates $a'_1 b'_1$ and $a''_1 b''_1$, and cooperatively generate the secret shares of $a''_1 b'_1 + a'_1 b''_2$ using the homomorphic Add-I and Mul-II properties of SHE; and matrix multiplications are essentially dot products of matrix rows and columns and therefore can be achieved by adopting the similar method.

D. Secure Matrix Inverse

The secure matrix inverse (SMI) protocol allows two parties to securely calculate the inverse of an additively secret-shared $n_1 \times n_1$ matrix, $\mathbf{C} = \mathbf{C}' + \mathbf{C}'' \bmod 2^l$. It is difficult to securely compute matrix inverse in ciphertexts due to the high complexity of matrix computations. To bypass complex ciphertext calculations, random unimodular matrices are introduced. The unimodular matrices have a unique property: the inverses of the matrices are also integer matrices. Therefore, random unimodular matrices can be proper randomness to randomize encrypted matrices with SHE. That is, an encrypted matrix can be first randomized by S_1, and then be shared with S_0 who decrypts to perform matrix inverse calculation. The inverse of the randomized matrix can be re-encrypted before returning to S_1 who can de-randomize the ciphertext using the homomorphic properties of SHE.

A random unimodular matrix can be created using the following steps: choose a random upper triangular matrix $\mathbf{B}_1$ and a random lower triangular matrix $\mathbf{B}_2$ (random numbers are chosen from $\mathbb{N}$); set main diagonal of $\mathbf{B}_1$ and $\mathbf{B}_2$ to ones; choose two random permutation matrices $\mathbf{W}_1$ and $\mathbf{W}_2$; and a random unimodular matrix is: $\mathbf{R} = \mathbf{P}_1\mathbf{B}_2\mathbf{B}_1\mathbf{P}_2$, as shown in Fig. 2. In our protocol, two random unimodular matrices, $\mathbf{R}_1$ and $\mathbf{R}_2$, are generated to randomize the matrix, $\mathbf{C}$, whose inverse needs to be calculated. Considering that the inverse of $\mathbf{C}$ may turn into a non-integer matrix, we also integrate an amplification parameter, η, to help round the non-integer matrix to an integer matrix with a little accuracy loss.

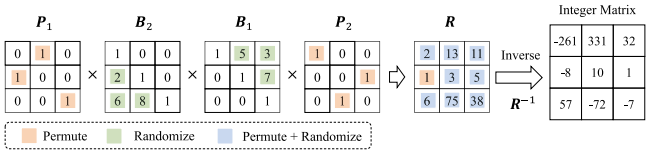
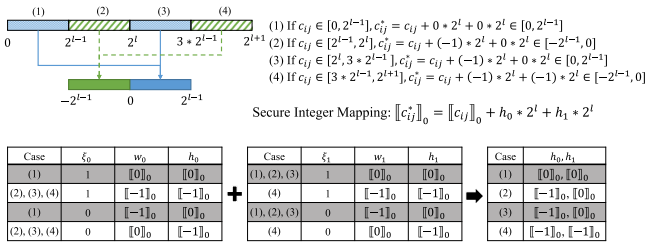


Fig. 2. Example of Generating a 3×3 Random Unimodular Matrix.



Case	ξ_0	w_0	h_0
(1)	1	$[\![0]\!]_0$	$[\![0]\!]_0$
(2), (3), (4)	1	$[\![-1]\!]_0$	$[\![-1]\!]_0$
(1)	0	$[\![-1]\!]_0$	$[\![0]\!]_0$
(2), (3), (4)	0	$[\![0]\!]_0$	$[\![-1]\!]_0$

+

Case	ξ_1	w_1	h_1
(1), (2), (3)	1	$[\![0]\!]_0$	$[\![0]\!]_0$
(4)	1	$[\![-1]\!]_0$	$[\![-1]\!]_0$
(1), (2), (3)	0	$[\![-1]\!]_0$	$[\![0]\!]_0$
(4)	0	$[\![0]\!]_0$	$[\![-1]\!]_0$

→

Case	h_0, h_1
(1)	$[\![0]\!]_0, [\![0]\!]_0$
(2)	$[\![-1]\!]_0, [\![0]\!]_0$
(3)	$[\![-1]\!]_0, [\![0]\!]_0$
(4)	$[\![-1]\!]_0, [\![-1]\!]_0$

Fig. 3. Securely Mapping $c_{ij} \in \mathbb{Z}_{2^l}$ into $c^*_{ij} \in \mathbb{Z}$.

Protocol 4: Secure Integer Mapping (SMAP).

Input: S_0:(sk_0); S_1:$([\![c_{ij}]\!]_0, \mathsf{pk}_0)$
Output: S_1:$[\![c^*_{ij}]\!]_0$
1: For $b \in \{0, 1\}$, S_1 flips a coin $\xi_b \in \{0, 1\}$, chooses two random numbers $r_{b,1}, r_{b,2} \in \mathbb{Z}_{2^l}$ ($r_{b,1} > r_{b,2}$), computes

$$\hat{\mathsf{ct}}_b = \begin{cases} r_{b,1} \cdot ([\![c_{ij}]\!]_0 - (1+2b) \cdot 2^{l-1}) \\ \quad + r_{b,2} & \text{if } \xi_b = 1; \\ r_{b,1} \cdot (-[\![c_{ij}]\!]_0 + 4 \cdot [\![0]\!]_0 \\ \quad + (1+2b) \cdot 2^{l-1}) - r_{b,2} & \text{if } \xi_b = 0, \end{cases} \tag{8}$$

and sends $\hat{\mathsf{ct}}_b$ to S_0
2: S_0 decrypts to obtain $\hat{m}_b = \mathsf{Dec}(\hat{\mathsf{ct}}_b)$. If $\hat{m}_b < \frac{L}{2}$, S_0 returns $w_b = [\![-1]\!]_0$; otherwise returns $w_b = [\![0]\!]_0$
3: S_1 calculates

$$h_b = \begin{cases} \xi_b - 1 + w_b & \text{if } \xi_b = 1; \\ \xi_b - 1 - w_b + 4 \cdot [\![0]\!]_0 & \text{if } \xi_b = 0, \end{cases} \tag{9}$$

and updates $[\![c_{ij}]\!]_0$ to $[\![c^*_{ij}]\!]_0 = [\![c_{ij}]\!]_0 + \sum_{b=\{0,1\}} h_b * 2^l$.

Moreover, if $[\![\mathbf{C}]\!]_0$ is created using homomorphic additions (without modulo operations), the plaintext of each encrypted entry $[\![c_{ij}]\!]_0$ ($i, j \in [1, n_1]$) of $[\![\mathbf{C}]\!]_0$ is in the field of $\mathbb{Z}_{2^{l+1}}$. When directly applying matrix randomization on $[\![\mathbf{C}]\!]_0$, the results of matrix inverse are not correct since the matrix inverse should be conducted in $\mathbb{Z}$. Accordingly, before randomizing $\mathbf{C}$, we propose a secure integer mapping (SMAP) protocol, which can convert $[\![c_{ij} \in \mathbb{Z}_{2^{l+1}}]\!]$ to its real value $[\![c^*_{ij} \in \mathbb{Z}]\!]$ without exposing c_{ij}. As shown in Fig. 3, we categorize possible mapping procedures into four cases, and construct a secure mapping function: $[\![c^*_{ij}]\!] = [\![c_{ij}]\!] + \sum_{b=\{0,1\}} h_b * 2^l$ to achieve the transformation. Through manipulating randomness $(\xi_b, r_{b,1}, r_{b,2})$ and utilizing the homomorphic properties of SHE, $h_{b\in\{0,1\}}$ can be generated without disclosing c_{ij}.

Finally, by integrating matrix randomization techniques and SMAP protocol, SMI protocol is given, and the details are described in Protocol 5. After running the protocol, S_0 gets $\mathbf{D}'$ and S_1 obtains $\mathbf{D}''$, satisfying $\mathbf{D} = \eta\mathbf{C}^{-1} = \mathbf{D}' + \mathbf{D}'' \bmod 2^l$.

Authorized licensed use limited to: University of Waterloo. Downloaded on February 17,2026 at 23:06:10 UTC from IEEE Xplore. Restrictions apply.

Protocol 5: Secure Matrix Inverse (SMI).

Input: S_0:($\mathbf{C}'$, sk_0); S_1:($\mathbf{C}''$, pk_0)
Output: S_0:$\mathbf{D}'$; S_1:$\mathbf{D}''$
1: S_0 encrypts $\mathbf{C}'$ as $[\![\mathbf{C}']\!]_0$ and shares $[\![\mathbf{C}']\!]_0$ with S_1
2: S_1 performs homomorphic additions: $[\![\mathbf{C}]\!]_0 = [\![\mathbf{C}']\!]_0 + \mathbf{C}''$
3: **for all** $i \in [1, n_1]$ and $j \in [1, n_1]$ **do**
4: S_0 and S_1 run $\mathtt{SMAP}([\![c_{ij}]\!]_0)$ to update each entry $[\![c_{ij}]\!]_0$ of $[\![\mathbf{C}]\!]_0$
5: **end for**
6: S_1 chooses two random unimodular matrices, $\mathbf{R}_1$ and $\mathbf{R}_2$, and shares $\mathbf{R}_1 \times [\![\mathbf{C}]\!]_0 \times \mathbf{R}_2 = [\![\mathbf{R}_1\mathbf{C}\mathbf{R}_2]\!]_0 + 2^{k_0-k_1-k_2-5} \cdot [\![0]\!]$ with S_0
7: S_0 decrypts to obtain $\mathbf{R}_1\mathbf{C}\mathbf{R}_2$, calculates the inverse as $\mathbf{R}_2^{-1}\mathbf{C}^{-1}\mathbf{R}_1^{-1}$, and sends $[\![\eta\mathbf{R}_2^{-1}\mathbf{C}^{-1}\mathbf{R}_1^{-1}]\!]_0 = [\![\mathbf{W}]\!]_0$ to S_1, where η is an amplification parameter
8: S_1 performs homomorphic multiplications: $\mathbf{R}_2 \times [\![\mathbf{W}]\!]_0 \times \mathbf{R}_1 = [\![\eta\mathbf{C}^{-1}]\!]_0 + 2^{k_0-k_1-k_2-5} \cdot [\![0]\!] = [\![\mathbf{D}]\!]_0$; generates an $n_1 \times n_1$ random matrix $\mathbf{R} = (r_{ij})_{i=1,j=1}^{n_1,n_1}$, where $r_{ij} \in \mathbb{Z}^l$ is a random number; and sets $\mathbf{D}'' = -\mathbf{R} \bmod 2^l$ and shares $[\![\mathbf{D}]\!]_0 + \mathbf{R}$ with S_0
9: S_0 decrypts $[\![\mathbf{D}]\!]_0 + \mathbf{R}$ to obtain $\mathbf{D} + \mathbf{R}$ and $\mathbf{D}' = \mathbf{D} + \mathbf{R} \bmod 2^l$

Protocol 6: Secure Integer Division (SIDIV).

Input: S_0:(α', ι, sk_0); S_1:(α'', ι, pk_0)
Output: S_0:ω'; S_1:ω''
1: S_0 shares $[\![\alpha']\!]_0$ with S_1
2: S_1 performs homomorphic additions: $\mathsf{ct} = [\![\alpha']\!]_0 + \alpha''$
3: S_0 and S_1 run $\mathtt{SMAP}(\mathsf{ct})$ to update ct
4: S_1 chooses a random number $r \in \mathbb{Z}_{2^l}$, sets $\omega'' = \lfloor -r/\iota \rfloor \bmod 2^l$, and shares $\mathsf{ret} = \mathsf{ct} + r$ with S_0
5: S_0 decrypts to obtain $\omega' = \lfloor \mathtt{Dec}(\mathsf{ret})/\iota \rfloor = \lfloor \frac{\alpha'+\alpha''+r}{\iota} \rfloor \bmod 2^l$

E. Secure Integer Division

With a public divisor, $\iota \in \mathbb{Z}$, the secure division protocol allows S_0 with the input of α' and S_1 with the input of α'' to securely compute $\lfloor \frac{\alpha}{\iota} \rfloor$, which is the floor rounding of $\frac{\alpha}{\iota}$. At the end of the protocol, S_0 obtains ω' and S_1 gets ω'', satisfying $\omega' + \omega'' \bmod 2^l = \lfloor \frac{\alpha}{\iota} \rfloor$. The rationale for designing the protocol is similar to SMI protocol: S_1 maps the plaintext of $[\![\alpha' + \alpha'']\!]_0$ into $\mathbb{Z}$ by cooperating with S_0, randomizes the ciphertext with a randomness $r \in \mathbb{Z}_{2^l}$ based on homomorphic properties of SHE, and shares $[\![\alpha' + \alpha'' + r]\!]_0$ with S_0; S_0 decrypts to perform the division operation on the randomized value. Finally, S_0 and S_1 can obtain $\omega' = \lfloor \frac{\alpha'+\alpha''+r}{\iota} \rfloor \bmod 2^l$ and $\omega'' = \lfloor \frac{-r}{\iota} \rfloor \bmod 2^l$, respectively. The detailed descriptions of the protocol are shown in Protocol 6.

F. Secure Absolute Value Difference

Suppose that S_0 has α' and β' and S_1 has α'' and β'', the secure absolute value difference (SAVD) protocol allows two parties to securely compute $\psi = \psi' + \psi'' \bmod 2^l = |\alpha - \beta|$, where ψ' is revealed to S_0 and ψ'' is known by S_1. The protocol is designed using SHE and SGET protocol, and its design principle is as follows: if $\alpha \geq \beta$, we have $\gamma = [\![1]\!]_0$, and thus $[\![\alpha - \beta]\!]_0 \cdot (2\gamma - 1) = [\![\alpha - \beta]\!]_0 \cdot [\![1]\!]_0 = [\![\alpha - \beta]\!]_0$. If $\alpha < \beta$, we have $\gamma = [\![0]\!]_0$, and therefore $[\![\alpha - \beta]\!]_0 \cdot (2\gamma - 1) = [\![\alpha - \beta]\!]_0 \cdot [\![-1]\!]_0 = [\![\beta - \alpha]\!]_0$. To avoid ciphertext multiplications, S_1 utilizes a random value, ξ, to conceal the real value of γ, and thus the protocol requires two communication rounds between S_0 and S_1. The detailed protocol is described in Protocol 7.

Protocol 7: Secure Absolute Value Difference (SAVD).

Input: S_0:(α', β', sk_0, pk_1); S_1:(α'', β'', pk_0, sk_1)
Output: S_0:ψ'; S_1:ψ''
1: S_0 and S_1 run $\mathtt{SGET}(\alpha', \beta'; \alpha'', \beta'')$ to obtain ϕ' and ϕ''
2: S_0 shares $[\![\phi']\!]_0$ and $[\![\alpha' - \beta']\!]_0$ with S_1
3: S_1 sets $\gamma = [\![\phi']\!]_0$ if $\phi'' = 0$ or sets $\gamma = 1 - \phi' + 4 \cdot [\![0]\!]_0$ otherwise; flips a coin $\xi \in [1, -1]$ and chooses a random number $r_1 \in \mathbb{Z}_{2^l}$; calculates

$$w = \begin{cases} [\![\alpha' - \beta']\!]_0 + \alpha'' - \beta'' + r_1 & \text{if } \xi = 1; \\ -[\![\alpha' - \beta']\!]_0 + 4 \cdot [\![0]\!]_0 - \alpha'' + \beta'' - r_1 & \text{if } \xi = -1, \end{cases} \tag{10}$$

and

$$h = \begin{cases} 2\gamma - 1 & \text{if } \xi = 1; \\ -2\gamma + 1 + 2^{k_0-k_1-k_2-5} \cdot [\![0]\!]_0 & \text{if } \xi = -1, \end{cases} \tag{11}$$

generates $[\![-\xi \cdot r_1]\!]_1$ and shares $(w, h, [\![-\xi \cdot r_1]\!]_1)$ with S_0
4: S_0 decrypts to obtain $\psi' = \mathtt{Dec}(w) * \mathtt{Dec}(h) - r_0 \bmod 2^l$, where $r_0 \in \mathbb{Z}_{2^l}$ is a random number, and sends z to S_1:

$$z = \begin{cases} [\![-\xi \cdot r_1]\!]_1 + r_0 & \text{if } \mathtt{Dec}(h) = 1; \\ -[\![-\xi \cdot r_1]\!]_1 + 4 \cdot [\![0]\!]_1 + r_0 & \text{if } \mathtt{Dec}(h) = -1 \end{cases} \tag{12}$$

5: S_1 decrypts to obtain $\psi'' = \mathtt{Dec}(z) \bmod 2^l$

G. Secure Integer Square Root

Suppose that S_0 has α' and S_1 has α'', the secure integer square root (SISR) protocol allows two parties to securely compute $\delta = \delta' + \delta'' \bmod 2^l = \lfloor \sqrt{\alpha} \rfloor \approx \sqrt{\alpha} = \sqrt{\alpha' + \alpha''}$ ($\alpha' + \alpha'' \bmod 2^l \in [0, 2^{l-1}]$). The details of the protocol are described in Protocol 8. Since we use an integer square root method, the result, i.e, $\delta = \delta' + \delta''$, is an approximation of the square root without the fractional part. The logic of designing such a protocol is to transform plaintext operations of an integer square root approach using bit-by-bit comparison [37] to ciphertext operations. Finding the integer square root of $\alpha \in [0, 2^{l-1}]$ intuitively is to locate a value $\delta \in [0, 2^{\lceil \frac{l}{2} \rceil}]$ that satisfies $\delta^2 \leq \alpha$ and $(\delta + 1)^2 > \alpha$. According to the principle, the square root algorithm works as follows: 1) initialize $\delta = 0$, and $\hat{l} = \lceil (l-1)/2 - 1 \rceil$; and 2) for $i = 0$ to $\lceil (l-1)/2 - 1 \rceil$, calculate $m = 2\delta + 2^{\hat{l}}$ and shift m left by $\hat{l} - i$ bits: if $\alpha \geq m$, calculate $\delta = \delta + 2^{\hat{l}}$ and $\alpha = \alpha - m$; and update $\hat{l} = \hat{l} - 1$. The challenge of transforming the plaintext algorithm to a ciphertext algorithm is to conceal the comparison result of α and m meanwhile updating α based on the result. We resolve the challenge by using SGET protocol and ASS, i.e., the update procedure becomes $\alpha = \alpha' + \alpha'' - (\gamma' + \gamma'') \bmod 2^l$, and we have

$$\gamma' + \gamma'' \bmod 2^l = \mathtt{Dec}(\chi) + z'\beta' + (z''\beta'' + r_1)$$
$$= z''\beta' + z'\beta'' - r_1 + z'\beta' + (z''\beta'' + r_1)$$

Authorized licensed use limited to: University of Waterloo. Downloaded on February 17,2026 at 23:06:10 UTC from IEEE Xplore. Restrictions apply.

Protocol 8: Secure Integer Square Root (SISR).

Input: S_0:(α', sk_0); S_1:$(\alpha'', \mathsf{pk}_0)$
Output: S_0:δ'; S_1:δ''
1: S_0 and S_1 initializes $\zeta = 2^{\lceil \frac{l-1}{2} - 1 \rceil}$
2: S_0 and S_1 set $w_0 = 0$ and $w_1 = 0$, respectively
3: **for** $i = 0$ to $\lceil \frac{l-1}{2} - 1 \rceil$ **do**
4: S_0 shares $c_0 = [\![w_0]\!]_0$ with S_1
5: S_1 calculates $c = (2(c_0 + w_1) + \zeta) * 2^{\lceil \frac{l-1}{2} - 1 - i \rceil}$, chooses a random number $\beta'' \in \mathbb{Z}_{2^l}$, and shares $c - \beta''$ with S_0
6: S_0 decrypts $c - \beta''$ to obtain $\beta' = \mathtt{Dec}(c - \beta'') \bmod 2^l$
7: S_0 and S_1 run $\mathtt{SGET}(\alpha', \beta'; \alpha'', \beta'')$ to obtain ϕ' and ϕ''
8: S_0 shares $[\![\phi']\!]_0$ with S_1
9: S_1 sets $\varphi = [\![\phi']\!]_0$ if $\phi'' = 0$ or sets $\varphi = 1 - [\![\phi'']\!]_0 + 4 \cdot [\![0]\!]_0$ otherwise; chooses $r'' \in \mathbb{Z}_{2^{l/2}}$ and updates $w_1 = w_1 - r'' \bmod 2^l$; chooses $z'' \in \mathbb{Z}_{2^{l-1}}$ and $r_1 \in \mathbb{Z}_{2^l}$; computes $\chi = (c - \beta'') \cdot z'' + (\varphi - z'') \cdot \beta'' - r_1$ and $\gamma'' = z''\beta'' + r_1 \bmod 2^l$; updates $\alpha'' = \alpha'' - \gamma'' \bmod 2^l$; and shares $(\varphi - z'', \chi, \varphi \cdot \zeta + r'')$ with S_0
10: S_0 decrypts to obtain $z' = \mathtt{Dec}(\varphi - z'') \bmod 2^l$, $\gamma' = \mathtt{Dec}(\chi) + z'\beta' \bmod 2^l$, and $r' = \mathtt{Dec}(\varphi \cdot \zeta + r'')$, and updates $\alpha' = \alpha' - \gamma' \bmod 2^l$ and $w_0 = w_0 + r' \bmod 2^l$
11: S_0 and S_1 update $\zeta = \zeta/2$
12: **end for**
13: S_0 stores $\delta' = w_0 \bmod 2^l$ and S_1 stores $\delta'' = w_1 \bmod 2^l$

$$= (z' + z'')(\beta' + \beta'') = \mathtt{Dec}(\varphi)(\beta' + \beta'')$$

$$= \begin{cases} 0 & \text{if } \mathtt{Dec}(\varphi) = 1; \\ \beta' + \beta'' & \text{if } \mathtt{Dec}(\varphi) = 0. \end{cases} \quad (13)$$

Note that, φ is the encrypted comparison result, and hence α is updated with disclosing any information to two parties.

H. Secure Oblivious Shuffle of Key-Value Pairs

The secure oblivious shuffle (SOS) protocol allows S_0 to input $(i, W'_i)_{i\in[1,N]}$ and a self-chosen random permutation $\pi_0(\cdot)$, and enables S_1 to input $(i, W''_i)_{i\in[1,N]}$ and a self-chosen random permutation $\pi_1(\cdot)$, where i is the key and W'_i (resp. W''_i) is the value. After the protocol execution, S_0 gets $(\epsilon'_i, \Omega'_i)_{i\in[1,N]}$ and S_1 obtains $(\epsilon''_i, \Omega''_i)_{i\in[1,N]}$, satisfying $\pi_0(\pi_1(i)) = \epsilon'_i + \epsilon''_i \bmod 2^l$ and $W'_{\pi_0(\pi_1(i))} + W''_{\pi_0(\pi_1(i))} = \Omega'_i + \Omega''_i \bmod 2^l$. The details of the protocol are shown in Protocol 9. The protocol can assure that neither the permutations nor the shuffled key-value pairs are exposed. From a high-level view, the shuffle is oblivious to two parties since S_0 (resp. S_1) independently introduces its own randomness $(\gamma^*_i, r^*_i)_{i\in[1,N]}$ (resp. $(\gamma_i, r_i)_{i\in[1,N]}$) to conceal the permutation $\pi_0(\cdot)$ (resp. $\pi_1(\cdot)$).

1) Secure Quicksort of Key-Value Pairs: The secure quicksort (SQ) protocol allows two parties to securely sort N secret-shared key-value pairs $((\epsilon'_i, \Omega'_i); (\epsilon''_i, \Omega''_i))_{i\in[1,N]}$, i.e., sorting the keys by values in ascending order without exposing the sorted keys and values. At the end of the protocol, S_0 obtains $(\theta'_1, \theta'_2, \ldots, \theta'_N)$ and S_1 gets $(\theta''_1, \theta''_2, \ldots, \theta''_N)$, satisfying

Protocol 9: Secure Oblivious Shuffle (SOS).

Input: S_0:$((i, W'_i)_{i\in[1,N]}, \pi_0, \mathsf{sk}_0, \mathsf{pk}_1)$; S_1:$((i, W''_i)_{i\in[1,N]}, \pi_1, \mathsf{sk}_1, \mathsf{pk}_0)$
Output: S_0:$(\epsilon'_i, \Omega'_i)_{i\in[1,N]}$; S_1:$(\epsilon''_i, \Omega''_i)_{i\in[1,N]}$
1: S_0 shares $([\![W'_i]\!]_0)_{i\in[1,N]}$ with S_1
2: **for** $i = 1$ to N **do**
3: S_1 performs homomorphic additions: $[\![W_i]\!]_0 = [\![W'_i]\!]_0 + W''_i$; chooses random numbers $\gamma_i, r_i \in \mathbb{Z}_{2^l}$; generates $\alpha_i = [\![i]\!]_0 + \gamma_i$ and $\beta_i = [\![W_i]\!]_0 + r_i$; and encrypts γ_i and r_i as $w_i = [\![\gamma_i]\!]_1$ and $h_i = [\![r_i]\!]_1$
4: **end for**
5: S_1 permutes $(\alpha_i, \beta_i, w_i, h_i)_{i\in[1,N]}$ to $(\bar{\alpha}_i, \bar{\beta}_i, \bar{w}_i, \bar{h}_i)_{i\in[1,N]}$ using π_1, and shares $(\bar{\alpha}_i, \bar{\beta}_i, \bar{w}_i, \bar{h}_i)_{i\in[1,N]}$ with S_0
6: S_0 permutes $(\bar{\alpha}_i, \bar{\beta}_i, \bar{w}_i, \bar{h}_i)_{i\in[1,N]}$ to $(\tilde{\alpha}_i, \tilde{\beta}_i, \tilde{w}_i, \tilde{h}_i)_{i\in[1,N]}$ using π_0
7: **for** $i = 1$ to N **do**
8: S_0 chooses random numbers $\gamma^*_i, r^*_i \in \mathbb{Z}_{2^l}$, and randomizes $\alpha^*_i = \tilde{\alpha}_i + \gamma^*_i$, $\beta^*_i = \tilde{\beta}_i + r^*_i$, $w^*_i = \tilde{w}_i + \gamma^*_i$, and $h^*_i = \tilde{h}_i + r^*_i$
9: **end for**
10: S_0 shares $(\alpha^*_i, \beta^*_i, w^*_i, h^*_i)_{i\in[1,N]}$ with S_1
11: **for** $i = 1$ to N **do**
12: S_1 decrypts w^*_i and h^*_i to obtain $\hat{w}_i$ and $\hat{h}_i$, and chooses random numbers $\epsilon''_i, \Omega''_i \in \mathbb{Z}_{2^l}$, and generates $D_i = \alpha^*_i - \hat{w}_i - \epsilon''_i$ and $E_i = \beta^*_i - \hat{h}_i - \Omega''_i$
13: **end for**
14: S_1 shares $(E_i, D_i)_{i\in[1,N]}$ with S_0
15: S_0 decrypts E_i and D_i to obtain $\epsilon'_i = \mathtt{Dec}(E_i) \bmod 2^l$ and $\Omega'_i = \mathtt{Dec}(D_i) \bmod 2^l$ for $i = 1$ to N

$\Omega_{\theta_1} < \Omega_{\theta_2} < \ldots < \Omega_{\theta_N}$. The design idea is similar to [38], but the difference is that we focus on key-value pairs and present the detailed protocol description in Protocol 10: with the assistance of SLT protocol, the quicksort algorithm for plaintexts is transformed to a secure two-party protocol for secret-shared key-value pairs. Since the quicksort algorithm is a recursion algorithm, the protocol is also recursively triggered by S_0 and S_1, and hence two recursive functions $\mathtt{Sort}(\cdot)$ and $\mathtt{Partition}(\cdot)$ are proposed in the protocol.

V. Scheme Design & Construction

In this section, we begin with a decentralized trust-based framework, which serves as the foundation for our proposed range-based positioning. Then, we construct a privacy-preserving range-based positioning scheme, named PPRP, that leverages the proposed secure sub-protocols.

A. Range-Based Positioning With Decentralized Trust

We first introduce a decentralized trust-based framework that relies on two LMF servers (e.g., two edge servers) to achieve privacy-preserving range-based positioning. In such a framework, UE and ACs do not need to spend extra local communication/computation costs to perturb or encrypt their range-based measurement data for preserving location privacy before reporting, and can simply split their data, $u_{i\in[0,N]}$, into two secret

Authorized licensed use limited to: University of Waterloo. Downloaded on February 17,2026 at 23:06:10 UTC from IEEE Xplore. Restrictions apply.

Protocol 10: Secure Quicksort (SQ).

Input: S_0:$((\epsilon'_i, \Omega'_i)_{i\in[1,N]}, \mathsf{sk}_0)$; S_1:$((\epsilon''_i, \Omega''_i)_{i\in[1,N]}, \mathsf{pk}_0)$
Output: S_0:$(\theta')_{i\in[1,N]}$; S_1:$(\theta'')_{i\in[1,N]}$
S_0 shares $([\![\epsilon'_i]\!]_0)_{i\in[1,N]}$ with S_1
S_1 performs additions: $[\![\epsilon_i]\!]_0 = [\![\epsilon'_i]\!]_0 + \epsilon''_i$ for $i = 1$ to N
S_0 and S_1 run $\mathtt{Sort}(([\![\epsilon_i]\!]_0)_{i\in[1,N]}, (\Omega'_i)_{i\in[1,N]}, (\Omega''_i)_{i\in[1,N]}, 1, N)$ to obtain $\mathbf{w}$
S_1 chooses random numbers $\theta'_i \in \mathbb{Z}_{2^l}$, calculates $E_i = \mathbf{w}[i] - \theta'_i$ for $i = 1$ to N, and shares $(E_i)_{i\in[1,N]}$ with S_0
S_0 decrypts E_i to obtain θ''_i for $i = 1$ to N
procedure $\mathtt{Sort}\mathbf{w}, \mathbf{h}', \mathbf{h}'', \mathsf{left}, \mathsf{right}$
  if $\mathsf{left} < \mathsf{right}$ **then**
    S_0 and S_1 runs $\mathtt{Partition}(\mathbf{w}, \mathbf{h}', \mathbf{h}'', \mathsf{left}, \mathsf{right})$ to obtain mid
    S_0 and S_1 runs $\mathtt{Sort}((\mathbf{w}, \mathbf{h}', \mathbf{h}'', \mathsf{left}, \mathsf{mid} - 1))$
    S_0 and S_1 runs $\mathtt{Sort}((\mathbf{w}, \mathbf{h}', \mathbf{h}'', \mathsf{mid} + 1, \mathsf{right}))$
  end if
  S_1 returns $\mathbf{w}$
end procedure
procedure $\mathtt{Partition}\mathbf{w}, \mathbf{h}', \mathbf{h}'', \mathsf{begin}, \mathsf{end}$
  S_1 sets $\mathsf{pivot}''_{\mathsf{val}} = \mathbf{h}''[\mathsf{end}]$
  S_0 sets $\mathsf{pivot}'_{\mathsf{val}} = \mathbf{h}'[\mathsf{end}]$
  S_0 and S_1 set $i = \mathsf{begin} - 1$
  for $j = \mathsf{begin}$ to $\mathsf{end} - 1$ **do**
    S_0 and S_1 run $\mathtt{SLT}(\mathbf{h}'[j], \mathsf{pivot}'_{\mathsf{val}}; \mathbf{h}''[j], \mathsf{pivot}''_{\mathsf{val}})$ to obtain ρ' and ρ'' and combine them to get $\rho = \rho' \oplus \rho''$
    if $\rho = 1$ **then**
      S_0 and S_1 accumulate $i = i + 1$
      S_1 swaps $\mathbf{w}[i]$ and $\mathbf{w}[j]$ and swaps $\mathbf{h}'[i]$ and $\mathbf{h}'[j]$
      S_0 swaps $\mathbf{h}''[i]$ and $\mathbf{h}''[j]$
    end if
  end for
  S_1 swaps $\mathbf{w}[i + 1]$ and $\mathbf{w}[\mathsf{end}]$ and swaps $\mathbf{h}'[i + 1]$ and $\mathbf{h}'[\mathsf{end}]$
  S_0 swaps $\mathbf{h}''[i + 1]$ and $\mathbf{h}''[\mathsf{end}]$
  S_0 and S_1 return $\mathsf{mid} = i + 1$
end procedure

shares using ASS, and report secret shares to LMF servers to achieve privacy-preserving measurement initialization.

The decentralized trust-based framework is illustrated in Fig. 4, where all computations for LoS/NLoS identification and location estimation are outsourced to the LMF servers. To construct a privacy-preserving LoS/NLoS identification protocol and a privacy-preserving location estimation protocol under the framework, we utilize the basic secure modules proposed in Section IV as building blocks.

B. Privacy-Preserving Range-Based Positioning

Following the system and the syntax defined in Section II, the proposed privacy-preserving range-based positioning scheme (PPRP) consists of three phases: 1) UE and ACs initialize range/range-difference measurement data (`Initialize` Algorithm); 2) LMF servers identify top-$\bar{N}$ possible LoS ACs (`Filter` Protocol), and 3) LMF servers perform location estimation and return the estimated location to UE (`Estimate`

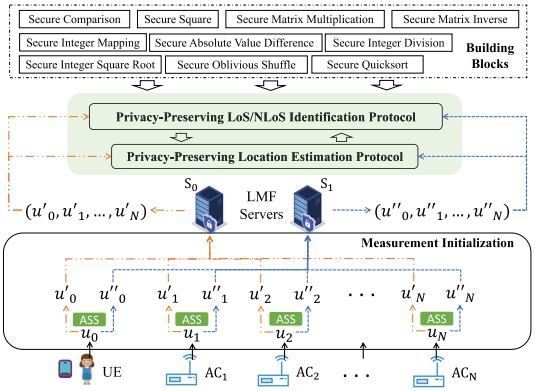


Fig. 4. Decentralized trust-based framework for range-based positioning.

Protocol). Under the decentralized framework, `Filter` protocol and `Estimate` protocol are designed on top of proposed building blocks, as shown in Fig. 5. The detailed descriptions of `Filter` protocol and `Estimate` protocol are given in Protocols 11 and 12.

- $\mathtt{Initialize}(u_i) \rightarrow (u'_i; u''_i)$: UE and N ACs first obtain range/range-difference measurement data, u_i for $i = 0$ to N. In DL and UL models, range/range-difference measurement data are different:
  1) DL Model (RM) - UE's range measurement data are $u_0 = (\tilde{d}_i)_{i\in[1,N]}$, and AC_i's measurement data are $u_i = (x_i, y_i)$ for $i = 1$ to N;
  2) DL Model (RDM) - UE's range-difference measurement data are $u_0 = (\tilde{d_{i1}})_{i\in[2,N]}$, and AC_i's measurement data are $u_i = (x_i, y_i)$ for $i = 1$ to N;
  3) UL Model (RM) - UE does not have range measurement data, and AC_i's measurement data are $u_i = (x_i, y_i, \tilde{d}_i)$ for $i = 1$ to N;
  4) UL Model (RDM) - UE does not have range-difference measurement data, AC_1's measurement data are $u_1 = (x_1, y_1)$, and AC_i's measurement data are $u_i = (x_i, y_i, \tilde{d_{i1}})$ for $i = 2$ to N.

  UE and ACs then report u'_i and u''_i ($u'_i + u''_i \bmod 2^l = u_i$) to S_0 and S_1, respectively: $\tilde{d}_i$ (resp. $\tilde{d_{i1}}$) can be split to $(\tilde{d_i}'; \tilde{d_i}'')$ using ASS, and x_i (resp. y_i) can be split to $(x'_i; x''_i)$ using ASS. To speed up the location estimation process, UE and ACs also upload the square values of u_i using ASS, which enables S_0 and S_1 to non-interactively construct secret-shared $\mathbf{A}$ and $\mathbf{b}$.
- $\mathtt{Filter}((u'_i)_{i=0}^N; (u''_i)_{i=0}^N; \bar{N}) \rightarrow ((v'_i)_{i=1}^{\bar{N}}; (v''_i)_{i=1}^{\bar{N}})$: To identify top-$\bar{N}$ possible LoS ACs from N ACs, S_0 and S_1 have to compute a secret-shared reference location at the beginning. The procedure of computing a secret-shared reference location is almost the same as the procedure of `Estimate` protocol except that S_0 and S_1 have to additionally run `SIDIV` protocol if the measurement data are range-difference measurement data (Recalling (3)). Here, we assume that the secret-shared reference location $((\hat{x}'_0, \hat{y}'_0); (\hat{x}''_0, \hat{y}''_0))$ can be obtained by S_0 and S_1, and discuss the procedure later. S_0 and S_1 then find all possible combinations of N ACs, i.e, $(\mathbb{S}_j)_{j\in[1,M]}$. For each subset $\mathbb{S}_j$, S_0 and S_1 collaboratively generate the secret-shared residuals RS'_j and RS''_j. Based on secret-shared residuals, each AC's secret-shared residuals (residual summation of subsets that the AC belong to) can be calculated by S_0 and S_1 using the homomorphic property

Authorized licensed use limited to: University of Waterloo. Downloaded on February 17,2026 at 23:06:10 UTC from IEEE Xplore. Restrictions apply.

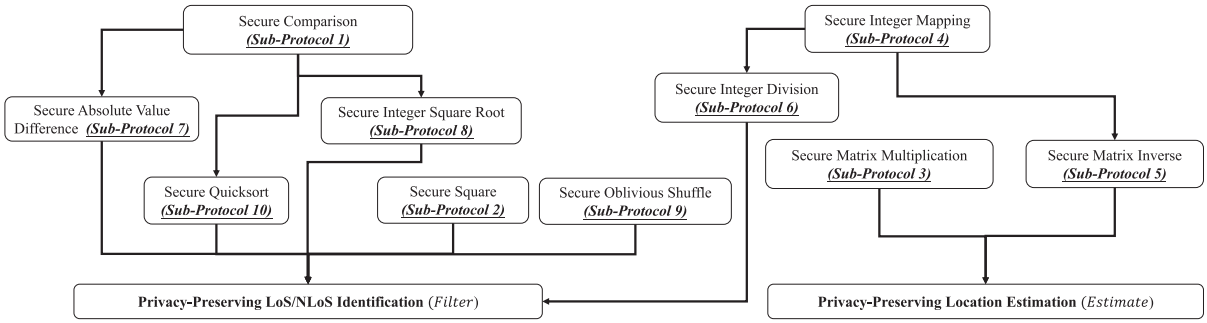


Fig. 5. Design overview: From basic modules to privacy-preserving los/nlos identification and location estimation.

of ASS. To rank ACs based on their residuals, S_0 and S_1 run SOS protocol and SQ protocol by inputting each AC's indexes and secret-shared residuals. The outputs are the secret-shared top-$\bar{N}$ possible LoS ACs' indexes (residuals are ordered in ascending order).

- $\texttt{Estimate}((u_i')_{i=0}^{N},(v_i')_{i=1}^{\bar{N}};(u_i'')_{i=0}^{N},(v_i'')_{i=1}^{\bar{N}};\mathsf{sign})\rightarrow(x_0,y_0)$*:* To estimate the position of UE, S_0 and S_1 first build the secret-shared matrix $\mathbf{A}'$ and $\mathbf{A}''$ and the secret-shared vector $\mathbf{b}'$ and $\mathbf{b}''$ (recalling (1) and (3)). Since the construction of $\mathbf{A}$ and $\mathbf{b}$ only involves addition/subtraction operations (based on (2) and (4)), S_0 and S_1 can independently assemble $\mathbf{A}'$ and $\mathbf{A}''$ based on the homomorphic property of ASS. With $((\mathbf{A}',\mathbf{b}');(\mathbf{A}'',\mathbf{b}''))$, S_0 and S_1 utilize SMM protocol and SMI protocol to get the secret shares of $\mathbf{Z}=\eta(\mathbf{A}^T\mathbf{A})^{-1}\mathbf{A}^T\mathbf{b}$, i.e., $\mathbf{Z}=\mathbf{Z}'+\mathbf{Z}''$. Finally, the estimated position of UE can be recovered at UE side after receiving $(\mathbf{Z}'[1],\mathbf{Z}'[2];\mathbf{Z}''[1],\mathbf{Z}''[2])$ from S_0 and S_1:

$$x_0=x_0'+x_0''=\begin{cases}\frac{1}{\eta}\mathbf{Z}'[1]+\frac{1}{\eta}\mathbf{Z}''[1] & (\mathrm{RM});\\ \frac{1}{2\eta}\mathbf{Z}'[1]+\frac{1}{2\eta}\mathbf{Z}''[1] & (\mathrm{RDM}),\end{cases}\tag{14}$$

and

$$y_0=y_0'+y_0''=\begin{cases}\frac{1}{\eta}\mathbf{Z}'[2]+\frac{1}{\eta}\mathbf{Z}''[2] & (\mathrm{RM});\\ \frac{1}{2\eta}\mathbf{Z}'[2]+\frac{1}{2\eta}\mathbf{Z}''[2] & (\mathrm{RDM}).\end{cases}\tag{15}$$

In case that UE sets $\mathsf{sign}=1$, top-$\bar{N}$ possible LoS ACs' indexes will be recovered: S_0 and S_1 reveal $v'_{i\in[1,\bar{N}]}$ and $v''_{i\in[1,\bar{N}]}$, and the measurement data received from top-$\bar{N}$ possible LoS ACs are utilized to build $(\mathbf{A}';\mathbf{A}'')$ and $(\mathbf{b}';\mathbf{b}'')$.

VI. Security Analysis

In this section, we first analyze the security of proposed building blocks, and then prove that the proposed scheme (PPRP) achieves location privacy using a simulation-based approach.

A. Security Analysis of Basic Modules

The proposed basic modules' security under the honest-cut-curious model is established on the security of SHE. Our security analysis adopts the following rationale: since $\mathcal{A}$ that controls either S_0 or S_1 will honestly follow the protocol, if there exists a simulator, $\mathcal{S}$, that does not know the inputs of the honest server (S_0 or S_1), can generate a view that is indistinguishable for $\mathcal{A}$, the security is guaranteed.

Security of SLT/SGET*:* ($\mathcal{A}$ *controls* S_1) $\mathcal{S}$ behaves as honest S_0 to simulate $[\![w']\!]_0$. The simulation can be easily achieved for the reason that $[\![w']\!]_0$ is the SHE ciphertext and SHE is IND-CPA secure. $\mathcal{A}$ cannot distinguish $[\![w']\!]_0$ from $[\![r]\!]_0$ where $r\in\mathcal{M}$ is a random number. ($\mathcal{A}$ *controls* S_0) $\mathcal{S}$ behaves as honest S_1 to simulate $\hat{\mathsf{ct}}$. If the comparison result $\phi=\phi'\oplus\phi''$ is revealed, $\mathcal{S}$ can adjust the plaintext of $\hat{\mathsf{ct}}$ to change ϕ' and correspondingly set ϕ'', satisfying $\phi'\oplus\phi''=\phi$. If ϕ is hidden, the simulation becomes easier: $\mathcal{S}$ can uniformly set the plaintext of $\hat{\mathsf{ct}}$ to make it positive or negative. Both simulations are indistinguishable from the view of $\mathcal{A}$.

Security of SS*:* ($\mathcal{A}$ *controls* S_1) $\mathcal{S}$ behaves as honest S_0 to simulate $[\![\alpha']\!]_0$. The simulation is straightforward due to the IND-CPA security of SHE. ($\mathcal{A}$ *controls* S_0) $\mathcal{S}$ behaves as honest S_1 to simulate $2\alpha''\cdot[\![\alpha']\!]_0+r$. Due to the fact that the randomness, $r\in\mathbb{Z}_{2^l}$, is introduced, $\mathcal{S}$ can simply generate a random SHE ciphertext to replace original messages. Both simulations are indistinguishable from the view of $\mathcal{A}$.

Security of SMM*:* ($\mathcal{A}$ *controls* S_1) $\mathcal{S}$ behaves as honest S_0 to simulate $[\![\mathbf{A}']\!]_0$ and $[\![\mathbf{B}']\!]_0$, which can be easily achieved considering that $[\![\mathbf{A}']\!]_0$ and $[\![\mathbf{B}']\!]_0$ are encrypted matrices. ($\mathcal{A}$ *controls* S_0) $\mathcal{S}$ behaves as honest S_1 to simulate $[\![\mathbf{C}]\!]_0-\mathbf{R}$, which is randomized by the random matrix, $\mathbf{R}$, chosen from $\mathcal{M}$. To complete the simulation, $\mathcal{S}$ can encrypt a random matrix as simulated messages. Both simulations are indistinguishable from the view of $\mathcal{A}$.

Security of SMAP*:* ($\mathcal{A}$ *controls* S_1) Since S_0 does not have private inputs, $\mathcal{S}$ just honestly follows the protocol to simulate S_0's operations. ($\mathcal{A}$ *controls* S_0) $\mathcal{S}$ behaves as honest S_1 to simulate ct_b for $b\in\{0,1\}$. In the protocol, three randomness numbers, $(r_{b,1},r_{b,2},\epsilon_b)$, are utilized to randomize the original plaintext of $[\![c_{ij}]\!]_0$, concealing $\hat{m}_b$. Therefore, $\mathcal{S}$ can substitute ct_b by a random SHE ciphertext to accomplish the simulation. Both simulations are indistinguishable from the view of $\mathcal{A}$.

Security of SMI*:* The security of SMI is partially based on the security of SMAP. As we have performed the simulation of SMAP, the simulations of relative procedures are ignored here. ($\mathcal{A}$ *controls* S_1) $\mathcal{S}$ behaves as honest S_0 to simulate $[\![\mathbf{C}']\!]_0$ and $[\![\mathbf{W}]\!]_0$. Due to the IND-CPA security of SHE, the ciphertexts can be simulated using random ciphertexts. ($\mathcal{A}$ *controls* S_0) $\mathcal{S}$ can generate encrypted random matrices to replace $[\![\mathbf{R}_1\mathbf{C}\mathbf{R}_2]\!]_0$ and use an encrypted random number to substitute $[\![\mathbf{D}]\!]_0+\mathbf{R}$ when simulating S_1. Both simulations are indistinguishable from the view of $\mathcal{A}$.

Security of SIDIV*:* Similar to SMI, the security of SIDIV is partially based on the security of SMAP, and thus we ignore redundant descriptions. ($\mathcal{A}$ *controls* S_1) $\mathcal{S}$ behaves as honest S_1

Authorized licensed use limited to: University of Waterloo. Downloaded on February 17,2026 at 23:06:10 UTC from IEEE Xplore. Restrictions apply.

Protocol 11: Privacy-Preserving Identification (`Filter`).

Input: S_0:$((u'_i)_{i=0}^{N}, \bar{N})$; S_1:$((u'_i)_{i=0}^{N}, \bar{N})$
Output: S_0:$(v'_i)_{i\in[1,\bar{N}]}$; S_1:$(v''_i)_{i\in[1,\bar{N}]}$
1: S_0 and S_1 partially run `Estimate` to obtain $\mathbf{Z}'$ and $\mathbf{Z}''$ (step-9)
2: **if** range measurements are applied **then**
3: S_0 and S_1 run $\texttt{SIDIV}(\mathbf{Z}'[1], \eta; \mathbf{Z}''[1], \eta)$ to obtain $\hat{x}'_0$ and $\hat{x}''_0$
4: S_0 and S_1 run $\texttt{SIDIV}(\mathbf{Z}'[2], \eta; \mathbf{Z}''[2], \eta)$ to obtain $\hat{y}'_0$ and $\hat{y}''_0$
5: **for** $i \in [1, N]$ **do**
6: S_0 and S_1 runs SS and SISR to obtain $\hat{d}'_i$ and $\hat{d}''_i$ ((5))
7: **end for**
8: **else if** range-difference measurements are applied **then**
9: S_0 and S_1 run $\texttt{SIDIV}(\mathbf{Z}'[1], 2\eta; \mathbf{Z}''[1], 2\eta)$ to obtain $\hat{x}'_0$ and $\hat{x}''_0$
10: S_0 and S_1 run $\texttt{SIDIV}(\mathbf{Z}'[2], 2\eta; \mathbf{Z}''[2], 2\eta)$ to obtain $\hat{y}'_0$ and $\hat{y}''_0$
11: **for** $i \in [2, N]$ **do**
12: S_0 and S_1 run SS, SISR, and SAVD to obtain $\hat{d}'_{i1}$ and $\hat{d}''_{i1}$ (6)
13: **end for**
14: **end if**
15: **for** $i \in [1, N]$ **do**
16: S_0 and S_1 initialize $W'_i = 0$ and $W''_i = 0$, respectively
17: **end for**
18: **for all** $\mathbb{S}_{j\in[1,M]}$ **do**
19: S_0 and S_1 run SS to obtain RS'_j and RS''_j
20: **for** $i \in [1, N]$ **do**
21: **if** $i \in \mathbb{S}_j$ **then**
22: S_0 locally updates $W'_i = W'_i + \mathsf{RS}'_j \bmod 2^l$
23: S_1 locally updates $W''_i = W''_i + \mathsf{RS}''_j \bmod 2^l$
24: **end if**
25: **end for**
26: **end for**
27: S_0 and S_1 run $\texttt{SOS}((i, W'_i)_{i\in[1,N]}; (i, W''_i)_{i\in[1,N]})$ to obtain $(\epsilon'_i, \Omega'_i)_{i\in[1,N]}$ and $(\epsilon''_i, \Omega''_i)_{i\in[1,N]}$
28: S_0 and S_1 run $\texttt{SQ}((\epsilon'_i, \Omega'_i)_{i\in[1,N]}; (\epsilon''_i, \Omega''_i)_{i\in[1,N]})$ to obtain $(\theta')_{i\in[1,N]}$ and $(\theta'')_{i\in[1,N]}$
29: S_0 and S_1 return $v'_i = \theta'_i$ and $v''_i = \theta''_i$ where $i \in [1, \bar{N}]$

Protocol 12: Privacy-Preserving Estimation (`Estimate`).

Input: S_0:$((u'_i)_{i=0}^{N}, (v'_i)_{i=1}^{\bar{N}}, \mathsf{sign})$;
S_1:$((u''_i)_{i=0}^{N}, (v''_i)_{i=1}^{\bar{N}}, \mathsf{sign})$
Output: UE:(x_0, y_0)
1: **if** $\mathsf{sign} = 1$ **then**
2: S_0 and S_1 first run $\texttt{Filter}((u'_i)_{i=0}^{N}; (u''_i)_{i=0}^{N}; \bar{N})$ to obtain $(v'_i)_{i=1}^{\bar{N}}$ and $(v''_i)_{i=1}^{\bar{N}}$ and then locally construct $(\mathbf{A}', \mathbf{b}')$ and $(\mathbf{A}'', \mathbf{b}'')$
3: **else**
4: S_0 and S_1 locally construct $(\mathbf{A}', \mathbf{b}')$ and $(\mathbf{A}'', \mathbf{b}'')$
5: **end if**
6: S_0 and S_1 run $\texttt{SMM}((\mathbf{A}')^T, \mathbf{A}'; (\mathbf{A}'')^T, \mathbf{A}'')$ to obtain $\mathbf{C}'$ and $\mathbf{C}''$
7: S_0 and S_1 run $\texttt{SMI}(\mathbf{C}'; \mathbf{C}'')$ to get $(\mathbf{D}'; \mathbf{D}'')$
8: S_0 and S_1 run $\texttt{SMM}((\mathbf{A}')^T, \mathbf{b}'; (\mathbf{A}'')^T, \mathbf{b}'')$ to obtain $\mathbf{E}'$ and $\mathbf{E}''$
9: S_0 and S_1 run $\texttt{SMM}(\mathbf{D}', \mathbf{E}'; \mathbf{D}'', \mathbf{E}'')$ to obtain $\mathbf{Z}'$ and $\mathbf{Z}''$
10: S_0 and S_1 send $\mathbf{Z}'$ and $\mathbf{Z}''$ to UE, respectively
11: UE recovers (x_0, y_0) according to (14) and (15)

to simulate $[\![\alpha']\!]_0$ by replacing it with arbitrary SHE ciphertexts. The simulation is perfect since $[\![\alpha']\!]_0$ and another SHE ciphertext are computationally indistinguishable. ($\mathcal{A}$ *controls* S_0) In the simulation, $\mathcal{S}$ encrypts a random number $r \in \mathbb{Z}_{2^l}$ as $\mathsf{ret} = [\![r]\!]_0$, and $\mathcal{A}$ cannot distinguish it if SHE is IND-CPA secure.

Security of `SAVD`*:* `SAVD` protocol utilizes `SGET` protocol as a submodule, and we omit the simulation of `SGET` because the composition is sequential. ($\mathcal{A}$ *controls* S_1) $\mathcal{S}$ behaves as honest S_0 to simulate $[\![\phi']\!]_0$, $[\![\alpha' - \beta']\!]_0$, and z. The simulations of $[\![\phi']\!]_0$ and $[\![\alpha' - \beta']\!]_0$ can be done obviously due to the IND-CPA security of SHE. Also, since z is protected by the randomness, $r \in \mathbb{Z}_{2^l}$, $\mathcal{S}$ can utilize randomness to generate an SHE ciphertext as an alternative in the simulation, which is indistinguishable. ($\mathcal{A}$ *controls* S_0) $\mathcal{S}$ behaves as honest S_1 to simulate w and h. Considering that r_1 and ϵ are randomly chosen in real-world executions, the simulation can be easily achieved, and $\mathcal{A}$ cannot distinguish the simulation from a real-world execution.

Security of `SISR`*:* Similar to `SAVD`, the security of `SISR` is partially based on the security of `SGET`, and we omit the simulation of this part. ($\mathcal{A}$ *controls* S_1) $\mathcal{S}$ behaves as honest S_0 to simulate $[\![\phi']\!]_0$. The simulation can be easily conducted and is indistinguishable since SHE is IND-CPA secure. ($\mathcal{A}$ *controls* S_0) $\mathcal{S}$ behaves as honest S_1 to simulate $c - \beta''$, $\rho - z''$, $\chi = (c - \beta'') \cdot z'' + (\varphi - z'') \cdot \beta'' - r_1$, and $\rho \cdot \zeta + r''$. Here, $c - \beta''$ is an SHE ciphertext whose plaintext is randomized by β'', and hence can be simulated by $\mathcal{S}$. Simulating $\rho - z''$, χ, and $\rho \cdot \zeta + r''$ can be achieved as well although $\mathcal{S}$ does not know ϕ'' and corresponding φ. The simulation is indistinguishable from the view of $\mathcal{A}$ due to the reason that φ is protected using three random numbers (r_1, z'', r'').

Security of `SOS`*:* ($\mathcal{A}$ *controls* S_1) $\mathcal{S}$ behaves as honest S_0 to simulate $([\![W'_i]\!]_0, \alpha^*_i, \beta^*_i, w^*_i, h^*_i)_{i\in[1,N]}$ where (α^*_i, β^*_i) are SHE ciphertexts encrypted by pk_0 and (w^*_i, h^*_i) are SHE ciphertexts encrypted by pk_1. Ciphertexts encrypted by pk_0 can be easily simulated as $\mathcal{A}$ cannot break the IND-CPA security of SHE, and ciphertexts encrypted by pk_1 can be simulated, taking into account that plaintexts are randomized by $(r^*_i)_{i\in[1,N]}$. ($\mathcal{A}$ *controls* S_0) $\mathcal{S}$ behaves as honest S_1 to simulate $(\bar{\alpha}_i, \bar{\beta}_i, \bar{w}_i, \bar{h}_i, E_i, D_i)_{i\in[1,N]}$, which are actually encrypted random numbers. As a result, $\mathcal{S}$ can employ the same logic as described above to carry out the simulation. Both simulations are indistinguishable from the view of $\mathcal{A}$.

Security of `SQ`*:* The security of `SAVD` is partially built upon the security of `SLT`. We have accomplished the simulation of `SLT`, and therefore we omit repetitive simulations. ($\mathcal{A}$ *controls* S_1) $\mathcal{S}$ behaves as S_0 to simulate $([\![\epsilon'_i]\!]_0)_{i\in[1,N]}$. In the simulation, $([\![\epsilon'_i]\!]_0)_{i\in[1,N]}$ can be replaced by arbitrary SHE ciphertexts. ($\mathcal{A}$ *controls* S_0) $\mathcal{S}$ behaves as S_0 to simulate $(E_i)_{i\in[1,N]}$. Note that, $(E_i)_{i\in[1,N]}$ are ciphertexts whose plaintexts are randomized by θ', and consequently they can be substituted by encrypted random numbers in the simulation. Both simulations are indistinguishable from $\mathcal{A}$'s view.

Authorized licensed use limited to: University of Waterloo. Downloaded on February 17,2026 at 23:06:10 UTC from IEEE Xplore. Restrictions apply.

B. Security Analysis of PPRP

We prove the security of PPRP based on a simulation-based real/ideal world model. Without loss of generality, we use $\mathbb{C} \subset [1, N]$ to denote the set of compromised ACs, and $\mathbb{H} \subset [1, N]$ to denote the set of honest ACs. The security of PPRP can be discussed in four cases:

- (*Case-I*) $\mathcal{A}$ corrupts S_0 and a set of ACs $\in \mathbb{C}$;
- (*Case-II*) $\mathcal{A}$ corrupts S_1 and a set of ACs $\in \mathbb{C}$;
- (*Case-III*) $\mathcal{A}$ corrupts S_0, a set of ACs $\in \mathbb{C}$, and UE;
- (*Case-IV*) $\mathcal{A}$ corrupts S_1, a set of ACs $\in \mathbb{C}$, and UE.

To make the proof concise, we analyze *Case-I* and *Case-III* together and discuss the security of *Case-II* and *Case-IV* simultaneously. This is because after the measurement initialization phase, the location estimation is primarily performed by S_0 and S_1, and the corruptions of ACs and UE only determine how much information is leaked to $\mathcal{A}$ at the beginning and the end of the protocols.

Simulation of Case-I and Case-III: We first define the leakages in these two cases based on the defined leakage function, $\mathcal{L}$. For *Case-I*, we have $\mathcal{L}^{\text{init}} = ((u_i)_{\text{AC}_i \in \mathbb{C}}, (u'_i)_{i \in N})$, $\mathcal{L}^{\text{filter}} = ((v'_i)_{i \in [1,N]}, \text{ir}^{\text{filter}})$, and $\mathcal{L}^{\text{est}} = ((v'_i)_{i \in [1,N]}, \mathbf{z}', \text{ir}^{\text{est}})$, where $\text{ir}^{\text{filter}}$ and ir^{est} are intermediate messages during the executions of `Filter` and `Estimate` protocols. If $\text{sign} = 1$, $\mathcal{L}^{\text{est}} = \mathcal{L}^{\text{est}} \cup (v_i)_{i \in [1,\bar{N}]}$. For *Case-III*, the estimated location of UE is also leaked, i.e., $\mathcal{L}^{\text{init}} = ((u_i)_{\text{AC}_i \in \mathbb{C}}, (u'_i)_{i \in N}, x_0, y_0)$. Based on the security of basic modules, the inputs and intermediate messages between $\mathcal{S}$ and $\mathcal{A}$ can be easily simulated in *Case-I* and *Case-III*, and the challenges are how to bias the outputs such that the simulated outputs and the real outputs have the same distribution. The idea is that $\mathcal{S}$ can behave as S_1 but modify the ciphertexts encrypted under S_0's public key (secret keys are controlled by $\mathcal{A}$). Since $\mathcal{S}$ has the public key of S_0, and the ciphertexts are randomized at S_1 side, $\mathcal{A}$ cannot distinguish the view generated by the simulation from the view generated in real-world executions. For example, with the leakages $(v_i)_{i \in [1,\bar{N}]}$, $\mathcal{S}$ can modify the step-27 in Protocol 11 (more specifically, the step-14 in Protocol 9), and use E'_i and D'_i to substitute the original E_i and D_i, where E'_i and D'_i are the ciphertexts of random numbers chosen by $\mathcal{S}$. These random numbers are stored at $\mathcal{S}$. When revealing $(v_i)_{i \in [1,\bar{N}]}$, since $\mathcal{S}$ knows $(v'_i)_{i \in [1,\bar{N}]}$, it can correspondingly change $(v''_i)_{i \in [1,\bar{N}]}$ to ensure $(v'_i)_{i \in [1,\bar{N}]} + (v'_i)_{i \in [1,\bar{N}]} = (v_i)_{i \in [1,\bar{N}]}$.

Simulation of Case-II and Case-IV: The leakages in these two cases are captured by the defined leakage function, $\mathcal{L}$. For *Case-II*, we have $\mathcal{L}^{\text{init}} = ((u_i)_{\text{AC}_i \in \mathbb{C}}, (u''_i)_{i \in N})$, $\mathcal{L}^{\text{filter}} = ((v''_i)_{i \in [1,N]}, \text{ir}^{\text{filter}})$, and $\mathcal{L}^{\text{est}} = ((v''_i)_{i \in [1,N]}, \mathbf{z}'', \text{ir}^{\text{est}})$, where $\text{ir}^{\text{filter}}$ and ir^{est} are intermediate messages during the protocol executions. If $\text{sign} = 1$, $\mathcal{L}^{\text{est}} = \mathcal{L}^{\text{est}} \cup (\theta_i)_{i \in [1,\bar{N}]}$. For *Case-IV*, $\mathcal{A}$ also knows (x_0, y_0). Similar to the simulation of *Case-I* and *Case-III*, we can straightforwardly construct a simulator, $\mathcal{S}$, to generate the view of the inputs and intermediate messages to $\mathcal{A}$, which is computationally indistinguishable. To bias the outputs, $\mathcal{S}$ can behave as S_0 to replace some intermediate messages with particular formats during the simulation. For example, with the leakages $(v_i)_{i \in [1,\bar{N}]}$, $\mathcal{S}$ can modify the step-27 in Protocol 11 (more specifically, the step-10 in Protocol 9) and use self-chosen ciphertexts to replace α_i^*, β_i^*, w_i^*, and h_i^* and stores the plaintexts corresponding to the ciphertexts. In this situation, $\mathcal{S}$ can recover $\epsilon'' = \texttt{SHE.Dec}(\alpha_i^*, \text{sk}_0) - \hat{w}_i - \texttt{SHE.Dec}(D_i, \text{sk}_0)$ and $\Omega''_i = \texttt{SHE.Dec}(\beta_i^*, \text{sk}_0) - \hat{h}_i - \texttt{SHE.Dec}(E_i, \text{sk}_0)$, and use the information to bias the outputs. Replacing α_i^*, β_i^*, w_i^*, and h_i^* will not be identified by $\mathcal{A}$, since these ciphertexts are permuted and randomized before sharing.

Theorem 2: If proposed basic modules are secure, the proposed positioning scheme, PPRP, is $\mathcal{L}$-semantically secure, i.e.,

$$\text{View}_{\mathcal{A}}^{\text{REAL}} \overset{c}{\approx} \text{View}_{\mathcal{A},\mathcal{S},\mathcal{L}}^{\text{IDEAL}}.$$

Proof Sketch: The theorem is proved by demonstrating that there exists a PPT simulator, $\mathcal{S}$, who behaves as honest S_0 or S_1 and other honest parties $\mathbb{H}$ to interact with $\mathcal{A}$ as an ideal simulation. The simulation of Π is built on the simulation of `SLT/SGET`, `SS`, `SMM`, `SMAP`, `SMI`, `SIDIV`, `SAVD`, `SISR`, `SOS`, and `SQ` protocols, namely, the inputs and intermediate messages between $\mathcal{S}$ and $\mathcal{A}$ can be simulated using $\mathcal{L}$, and cannot be distinguished by $\mathcal{A}$. In addition, $\mathcal{S}$ can bias the outputs of the simulation to ensure that the simulated outputs have the same distribution as the outputs of real-world executions.

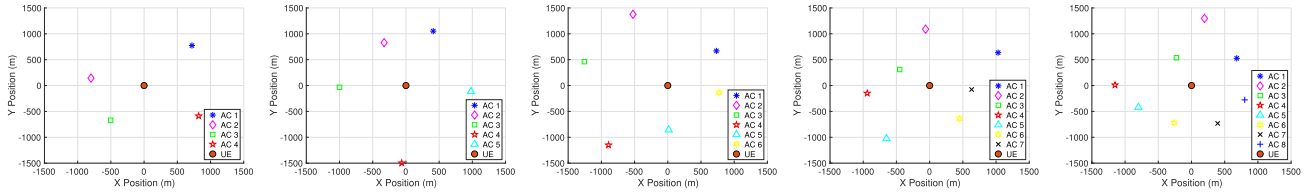
Under the condition that PPRP is $\mathcal{L}$-semantically secure, we discuss two attacks on PPRP defined in the adversarial model (Section II). For *Attack-I*, PPRP can guarantee that the exact locations of honest and uncompromised ACs cannot be revealed by solving the polynomial equations defined in (1) and (3), provided that there are more than one such ACs. Adversaries, in both the downlink (DL) and uplink (UL) models, cannot obtain more than three polynomial equations, while there exist more than three unknown variables. For *Attack-II*, PPRP can perfectly resist such an attack in DL model. However, in the UL model, it is inevitable that three or more ACs may collude to recover the location of the user equipment (UE).

VII. Performance Evaluation

This section shows the performance of the proposed privacy-preserving range-based positioning scheme (PPRP) in terms of four utilities defined in Section II-C. To clearly demonstrate the performance evaluation results, we also choose four state-of-the-art schemes as baselines and compare their performance with our work:

- *PHE-based Privacy-Preserving Positioning (PHEPP) [21]*: The first baseline is Alanwar et al.'s scheme constructed using partially homomorphic encryption (PHE). PHE cannot be straightforwardly applied to achieve secure computations defined (1), since PHE does not support homomorphic multiplication of ciphertexts. To resolve the challenge, a polyhedra-based approximation approach is presented to avoid multiplications of ciphertexts when performing location estimation.
- *PPS-based Privacy-Preserving Positioning (PPSPP) [17]*: The second baseline is Shi et al.'s scheme designed based on privacy-preserving summation (PPS). In their scheme, UE and N ACs can cooperatively embed self-chosen randomness into measurement data, and the noised values are aggregated at the home AC and UE finally. Due to PPS, the random numbers embedded in measurement data can be cancelled with each other, and thus $\mathbf{A}^T\mathbf{A}$ and $\mathbf{A}^T\mathbf{b}$ can be recovered. According to (3), with $\mathbf{A}^T\mathbf{A}$ and $\mathbf{A}^T\mathbf{b}$, UE can make simple computations to obtain (x_0, y_0).
- *PHE-assisted PPSPP (PHE+PPS) [20]*: The third baseline is one of the most classical privacy-preserving positioning

Authorized licensed use limited to: University of Waterloo. Downloaded on February 17,2026 at 23:06:10 UTC from IEEE Xplore. Restrictions apply.



(a) 4 ACs ($N = 4$) (b) 5 ACs ($N = 5$) (c) 6 ACs ($N = 6$) (d) 7 ACs ($N = 7$) (e) 8 ACs ($N = 8$)

Fig. 6. Positions of UE and N ACs randomly deployed in a $3,000 * 3,000\ m^2$ square area.

schemes proposed by Shu et al. Their scheme defines three privacy levels for range-based positioning, and we adopt the third protocol proposed in their scheme which is claimed to achieve level-III privacy.

- *FHE-based Baseline (FHEBASE) [22]*: We use CKKS, the most efficient fully homomorphic encryption (FHE) for arithmetic on approximate numbers, to design a baseline that achieves privacy-preserving range-based positioning and NLoS/LoS identification based on residual analysis. Although FHE can support complex operations, the baseline also requires a two-server model to achieve non-linear functions such as secure comparison and matrix inverse. Moreover, since bootstrapping is extremely time-consuming, the baseline relies on the two-server model to achieve fast bootstrapping in some cases, e.g., computing square root with Newton's iteration.

These four baselines are representative since they have different emphases. PHEPP is mainly designed for DL model with range measurements (RM) while PPSPP is primarily suitable for DL and UL model with range-difference measurements (RDM). Different from them, PHE+PPS is particularly applied in UL model with RM, while FHEBASE can be applied in both DL and UL models with either RM or RDM. They also have different privacy guarantees: since $\mathbf{A}^T\mathbf{A}$ and $\mathbf{A}^T\mathbf{b}$ are exposed during the location estimation process, PPSPP and PPS+PHE need stronger security assumptions to ensure privacy requirements, while PHEPP and FHEBASE rely on IND-CPA-secure encryptions and have a stronger security model like PPRP.

Experiment Setting: We develop a proof-of-concept prototype using Python for PPRP and the baselines. The source code of the prototype is posted on GitHub [39]. Based on the prototype, we conduct experiments on a desktop equipped with an M1 chip and 16 GB of RAM. Our experiments simulate real-world scenarios: In a square area of $3,000 \times 3,000\ \mathrm{m}^2$, $N = \{4, 5, 6, 7, 8\}$ ACs are randomly deployed, and one AC (other than the home AC) is chosen as the NLoS AC, as illustrated in Fig. 6. The experiments do not take into account the transmission delay between ACs and the UE; they measure only the execution delay to denote the computational costs. Moreover, the experiments are conducted on a single thread.

Public Parameters: The paper does not focus on improving range/range-difference measurement accuracy, and hence we set the LoS bias as 5 m and the NLoS bias as 300 m. For the security parameters of SHE, we set $k_0 = 500$, $k_1 = 200$, $k_2 = 100$, and $l = 95$. In such a setting, we have $t = 1,245$, i.e., at most 1,245 ciphertexts can be revealed. Additionally, we choose a security parameter of 2048 bits for Paillier encryption and configure CKKS with a scaling factor of 2^{45} (a scaling factor of 2^{40} will cause errors), a polynomial degree of 16 and a ciphertext modulus of 600 bits. The security parameter for PPS is 32 bits, i.e., random numbers for perturbing measurement data are 32 bits.

Amplification Parameter Chosen: In the implementation of Protocol 5 for secure matrix inversion, the amplification parameter η is critical in controlling the trade-off between result accuracy and computational efficiency. While a larger η helps reduce the inevitable accuracy loss caused by converting irreducible fractions of an inverted matrix to integers, the amplification must be contained within the decryptable domain of somewhat homomorphic encryption (SHE), which is determined by the security parameter k_1. Increasing η leads to a corresponding increase in k_1 and k_0, causing higher computational overheads as shown in Table II. Hence, for our experimental setup with ACs and UEs in a $3000 \times 3000\ \mathrm{m}^2$ area, $\eta = 10^{20}$ has been chosen to balance matrix inversion accuracy and SHE's computational overheads.

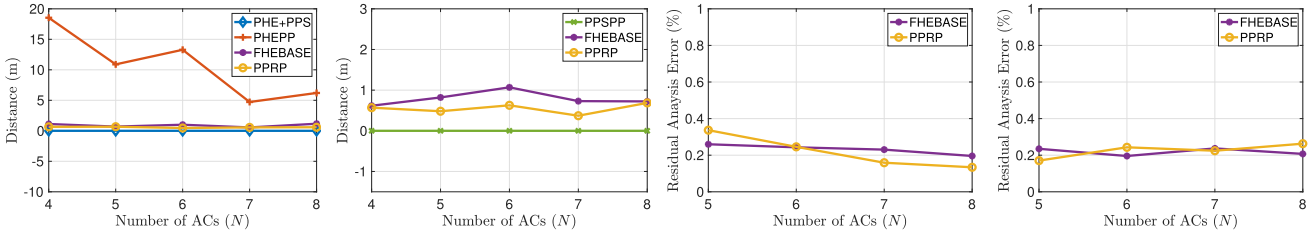
A. Errors Caused by Privacy-Preserving Techniques

One of the primary goals of a privacy-preserving range-based positioning scheme is to ensure that it achieves accuracy that is similar to the plaintext positioning schemes [28]. To assess the positioning errors arising from the use of privacy-preserving techniques, we compare the position errors of PPRP with the baselines in Fig. 7(a) and (b). Specifically, we utilize the euclidean distance between UE's estimated location obtained using the plaintext positioning schemes and UE's estimated location obtained using PPRP or the baselines to measure the positioning errors.

PPRP and FHEBASE introduce a few positioning errors compared to PPSPP. For instance, PPRP estimates UE's location as (1359.0124,1693.3680), while the plaintext positioning scheme (RM) estimates it as $(1358.9311, 1693.4519)$. This is due to the fact that PPRP and FHEBASE require integers as inputs for SHE and FHE, and thus the inputs and generated intermediate results are rounded off, thereby removing the decimal parts. In comparison to PHEPP, PPRP can preserve higher accuracy. PHEPP necessitates ACs to conduct sampling operations before location estimation using the polyhedra-based approximation method [21]. For maintaining high location accuracy, numerous sampling points are required, and we employed 100 sampling points in our experiment. Nevertheless, employing more sampling points will result in higher computation overheads during the `Initialize` phase at the AC side (DL model). Additionally, PHEPP suffers from geometric dilution issues that cause estimation errors, which may adversely affect the location accuracy.

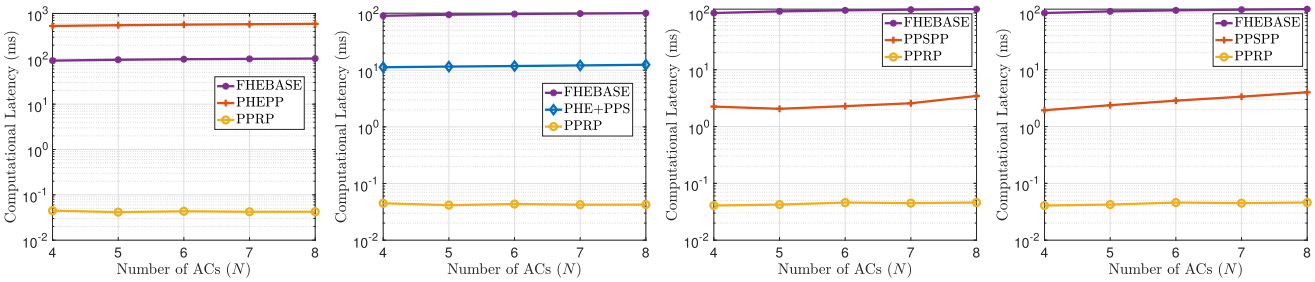
Furthermore, we conduct an evaluation of the residual analysis errors in privacy-preserving NLoS/LoS identification, and the experimental results are presented in Fig. 7(c) and (d). The residual analysis errors are represented by $\mathsf{RE} = \frac{1}{N}\sum_{i=0}^{N}\frac{|W_i - \hat{W}_i|}{W_i}$,

Authorized licensed use limited to: University of Waterloo. Downloaded on February 17,2026 at 23:06:10 UTC from IEEE Xplore. Restrictions apply.



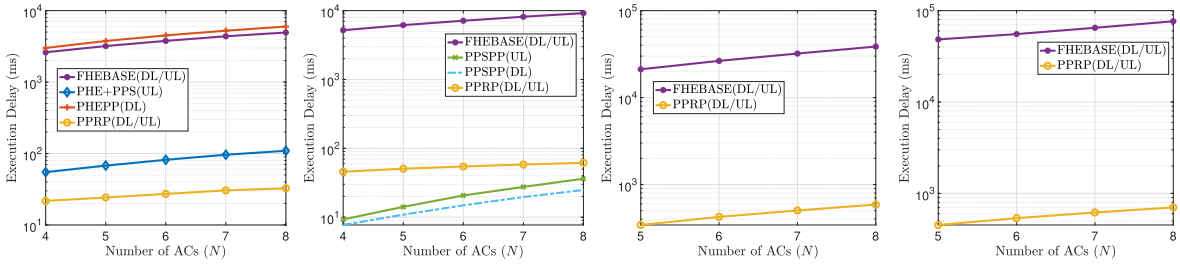
(a) Positioning Error (RM) (b) Positioning Error (RDM) (c) Residual Error (RM) (d) Residual Error (RDM)

Fig. 7. Positioning errors and residual analysis errors: Comparisons in DL/UL models with range/range-difference measurements (RM or RDM).



(a) Downlink Model (RM) (b) Uplink Model (RM) (c) Downlink Model (RDM) (d) Uplink Model (RDM)

Fig. 8. Amortized UE/AC's computational costs in DL/UL models with range/range-difference measurements (RM or RDM).



(a) `Estimate` (RM) (b) `Estimate` (RDM) (c) `Filter` (RM) (d) `Filter` (RDM)

Fig. 9. End-to-end execution delay of privacy-preserving range-based positioning (`Estimate`) and LoS/NLoS identification (`Filter`) with range/range-difference measurements (RM or RDM).

where W_i is AC$_i$'s residual generated by the plaintext identification schemes [24], [25], and $\hat{W}_i$ is AC$_i$'s residual generated by PPRP or FHEBASE. The results indicate that PPRP and FHEBASE have small residual errors, and both of them can guarantee the same NLoS/LoS identification rates as the plaintext schemes.

B. Computational Costs of UE and ACs

We evaluate the computational costs of UE and ACs in privacy-preserving range-based positioning, and the comparison results between PPRP and the baselines are illustrated in Fig. 8. PPRP outperforms all baselines apparently, i.e., it takes less than 0.1 ms for UE and ACs to report measurement data to LMF servers in both DL and UL models. In PHEPP, ACs are required to encrypt all sampling points, which incurs linear computational costs proportional to the number of sampling points and is time-consuming accordingly. In FHEBASE, UE and ACs must encrypt their measurement data using the CKKS encryption algorithm, which is even more computationally expensive than the Paillier encryption algorithm. Although PHE+PPS and PPS are more efficient than PHEPP and FHEBASE, they require UE and ACs to not only perform measurement initialization but also participate in the location estimation process. Consequently, these schemes still require significant computational resources. In contrast, PPRP only necessitates UE and ACs to conduct lightweight secret sharing operations and then go offline after uploading the secret-shared measurement data, rendering it highly computationally efficient.

C. End-to-End Execution Delay

We evaluate the end-to-end execution delay of PPRP and the baselines, without considering transmission delay. The results, averaged over 100 iterations, are shown in Fig. 9. Our evaluation focuses on two privacy-preserving protocols: location estimation protocol (`Estimate`) and NLoS/LoS identification protocol (`Filter`). The evaluation results are presented according to two

Authorized licensed use limited to: University of Waterloo. Downloaded on February 17,2026 at 23:06:10 UTC from IEEE Xplore. Restrictions apply.

categories (i.e., two measurement data generated by UE and ACs).

Fig. 9(a) demonstrates that PPRP outperforms PHEPP, FHEBASE, and PHE+PPS in terms of execution latency. Specifically, with 8 ACs deployed, PPRP completes `Estimate` in less than 35 ms, whereas PHEPP takes over 5,900 ms (with 100 points sampled) and FHEBASE requires more than 4,900 ms. Though PHE+PPS is more efficient than PHEPP and FHEBASE, it still takes 100 ms to complete the location estimation and requires all UEs and ACs to be online during the estimation. In Fig. 9(b), PPSPP is shown to be more efficient than PPRP, although PPRP takes less than 65 ms to complete the estimation process. PPSPP is more efficient because it allows revealing intermediate results during the location estimation and can bypass complex matrix operations such as matrix inverse. In other words, it sacrifices security for improved efficiency.

Fig. 9(c) and (d) depict the execution delay of `Filter`. Since PHEPP, PHE+PPS, and PPSPP do not focus on NLoS/LoS identification, we mainly compare PPRP with FHEBASE that achieves privacy-preserving NLoS/LoS identification using straightforward homomorphic operations. The results show that PPRP is more computation-efficient and can meet the real-world deployment requirements. For example, it takes less than 800 ms to identify possible NLoS ACs out of 8 ACs and spends less than 900 ms to sequentially execute `Estimate` and `Filter`.

D. Communication Complexity

In terms of communication rounds, PPRP requires $O(N)$ rounds for communication between UE/ACs and LMF servers in both DL and UL models. This is consistent with the communication rounds required by PHEPP and FHEBASE. On the other hand, PPSPP and PHE+PPS require $O(N^2)$ communication rounds, considering that UE and ACs need to transmit the randomness to perturb their measurement data. In pervasive environments, UE and the home AC play the role of LMF servers, therefore increasing the communication overheads.

VIII. Related Works

This section mainly reviews recent privacy-preserving range-based positioning schemes for mobile networks [14], [15], [16], [17], [18], [19], [20], [21].

From a high-level perspective, range-based positioning methods measure location dependent parameters such as time of arrival (ToA), time difference of arrival (TDoA), and received signal strength indicator (RSS) in mobile networks, and with these parameters, the distance among anchors (ACs) and user equipment (UE) can be calculated, and be further combined with the locations of ACs to estimate the location of UE. To protect not only UE's but also ACs' location privacy, many privacy-preserving range-based positioning schemes have been proposed. In these schemes, the location estimation problems can be transformed to least-square optimization problems, and we can roughly categorize these schemes into : 1) schemes based on *SR-LS* that fully uses range measurements between three or more ACs with UE [14], [16], [18], [19], [20], [21], and 2) schemes based on *SRD-LS* that depends on range-difference measurements between three or more ACs with UEs [15], [17], [40], [41]. For the first category, traditional secure multi-party computation techniques (SMC) such as garbled circuits and oblivious transfer are utilized by Hussain et al. to propose privacy-preserving positioning method for automotive systems [19]. Similar to SMC, fully homomorphic encryption (FHE) is also workable, but the FHE-based privacy-preserving positioning needs to tolerate more computational and communication costs [22]. To get rid of FHE's high complexity, relatively computation-efficient partially homomorphic encryption (PHE) such as Paillier encryption is applied. Under the condition where the locations of all ACs are publicly known, Jiang et al. proposed a PHE-based privacy-preserving positioning scheme in mobile edge computing scenarios [18]. To provide stronger privacy protection by hiding private ACs' locations, Alanwar et al. proposed a PHE-based privacy-preserving positioning scheme by using a polyhedra-based transformation, at the expense of sacrificing positioning accuracy caused by geometric dilution [21].

Unlike traditional cryptographic techniques, another research branch turns to using more lightweight privacy-preserving summation (PPS) techniques. The basic idea to achieve PPS-based privacy-preserving positioning is to conceal sensitive measurement data by randomness generated by UE and ACs. The randomness is generated in a special format such that random numbers generated by all participants can be cancelled after summation/aggregation. Shu et al. first proposed the idea and design their schemes with three privacy levels [20]. To realize the highest privacy guarantee (Level-III privacy), they have to combine PPS with PHE. Also, their scheme is particularly designed for pervasive environments, i.e., one of the ACs should be responsible for the final calculations. Different from Shu et al.'s work, Liu et al. redesigned the randomness distribution algorithm. In this way, PHE is not necessary, but their scheme only works with three ACs (2D positioning) or four ACs (3D positioning), namely, their scheme is a privacy-preserving trilateration-based positioning scheme [16]. Recently, Shi et al. proposed the first privacy-preserving range-based positioning scheme where UEs and ACs' inputs are range-different measurement data [17]. Their scheme requires the home AC and UE to aggregate other ACs' and UE's inputs. Same as Shu et al.'s work [20], their scheme works in pervasive environments. Following their work, Zhu et al. proposed a variant privacy-preserving multilateration scheme that uses the homomorphic property of Shamir's secret sharing to construct PPS [14]. To bypass complicated privacy-preserving protocol designs, Yan et al. proposed to use secure hardware such as Intel SGX to simplify the procedures of protecting location privacy for range-based positioning [15]. The security of their scheme is fully dependent on the security of Intel SGX.

In addition to privacy-preserving range-based positioning schemes, there exist other privacy-preserving positioning schemes that rely on fingerprint-based methods [42], and secure positioning schemes that focus on handling system-level security issues, e.g., ACs' location verification, in mobile networks [43]. We omit them since these works are beyond the scope of the paper.

IX. Conclusion

We have proposed a privacy-preserving range-based positioning scheme for mobile networks, named PPRP, which is built upon a decentralized trust-based framework. Under the framework, a set of secure two-party computation sub-protocols have been developed based on lightweight additive secret sharing and somewhat homomorphic encryption to achieve many fundamental functions such as matrix multiplication, matrix

Authorized licensed use limited to: University of Waterloo. Downloaded on February 17,2026 at 23:06:10 UTC from IEEE Xplore. Restrictions apply.

inverse, comparison, division, oblivious shuffle and sorting. These sub-protocols not only are the building blocks of our scheme to achieve location privacy protection during the location estimation and LoS/NLoS analysis, but also may be utilized to construct other secure schemes. A proof-of-concept prototype has also been developed to demonstrate the feasibility and practicality of our scheme. For the future work, we will investigate other positioning methods such as fingerprinting-based methods and aim to design a privacy-preserving fingerprinting-based positioning scheme under the decentralized framework.

REFERENCES

[1] Z. Li, Z. Zheng, S. Guo, B. Guo, F. Xiao, and K. Ren, "Disguised as privacy: Data poisoning attacks against differentially private crowdsensing systems," *IEEE Trans. Mobile Comput.*, vol. 22, no. 9, pp. 5155–5169, Sep. 2023.

[2] C. Huang, W. Wang, D. Liu, R. Lu, and X. Shen, "Blockchain-assisted personalized car insurance with privacy preservation and fraud resistance," *IEEE Trans. Veh. Technol.*, vol. 72, no. 3, pp. 3777–3792, Mar. 2023.

[3] Q. Kong, R. Lu, F. Yin, and S. Cui, "Blockchain-based privacy-preserving driver monitoring for MaaS in the vehicular IoT," *IEEE Trans. Veh. Technol.*, vol. 70, no. 4, pp. 3788–3799, Apr. 2021.

[4] H. Chen and H. Wymeersch, "Phone signals can help you find your way in cities even without GPS," 2022. [Online]. Available: https://www.nature.com/articles/d41586--022-03696-3

[5] J. Shen, J. Y. Won, Z. Chen, and Q. A. Chen, "Drift with devil: Security of multi-sensor fusion based localization in high-level autonomous driving under GPS spoofing," in *Proc. USENIX Secur. Symp.*, S. Capkun and F. Roesner, Eds, 2020, pp. 931–948.

[6] F. Tong, B. Ding, Y. Zhang, S. He, and Y. Peng, "A single-anchor mobile localization scheme," *IEEE Trans. Mobile Comput.*, vol. 23, no. 1, pp. 56–69, Jan. 2024.

[7] L. L. de Oliveira, G. H. Eisenkraemer, E. A. Carara, J. B. Martins, and J. Monteiro, "Mobile localization techniques for wireless sensor networks: Survey and recommendations," *ACM Trans. Sensor Netw.*, vol. 19, pp. 1–39, 2023.

[8] Y. Zhao, Z. Li, N. Cheng, W. Wang, C. Li, and X. Shen, "Covert localization in wireless networks: Feasibility and performance analysis," *IEEE Trans. Wireless Commun.*, vol. 19, no. 10, pp. 6549–6563, Oct. 2020.

[9] C. Zhang, L. Zhu, C. Xu, J. Ni, C. Huang, and X. Shen, "Location privacy-preserving task recommendation with geometric range query in mobile crowdsensing," *IEEE Trans. Mobile Comput.*, vol. 21, no. 12, pp. 4410–4425, Dec. 2022.

[10] V. Sadhu, S. Zonouz, V. Sritapan, and D. Pompili, "CollabLoc: Privacy-preserving multi-modal collaborative mobile phone localization," *IEEE Trans. Mobile Comput.*, vol. 20, no. 1, pp. 104–116, Jan. 2021.

[11] Q. Shi, X. Cui, S. Zhao, and M. Lu, "Sequential TOA-based moving target localization in multi-agent networks," *IEEE Commun. Lett.*, vol. 24, no. 8, pp. 1719–1723, Aug. 2020.

[12] M. Atif, R. Ahmad, W. Ahmad, L. Zhao, and J. J. Rodrigues, "UAV-assisted wireless localization for search and rescue," *IEEE Syst. J.*, vol. 15, no. 3, pp. 3261–3272, Sep. 2021.

[13] F. Ye and S. El Rouayheb, "Intermittent private information retrieval with application to location privacy," *IEEE J. Sel. Areas Commun.*, vol. 40, no. 3, pp. 927–939, Mar. 2022.

[14] Y. Zhu and J. Hu, "To hide anchor's position in range-based wireless localization via secret sharing," *IEEE Wireless Commun. Lett.*, vol. 11, no. 7, pp. 1325–1328, Jul. 2022.

[15] Z. Yan, X. Qian, S. Liu, and R. Deng, "Privacy protection in 5G positioning and location-based services based on SGX," *ACM Trans. Sensor Netw.*, vol. 18, no. 3, pp. 1–19, 2022.

[16] S. Liu and Z. Yan, "Efficient privacy protection protocols for 5G enabled positioning in Industrial IoT," *IEEE Internet Things J.*, vol. 9, no. 19, pp. 18527–18538, Oct. 2022.

[17] X. Shi and J. Wu, "To hide private position information in localization using time difference of arrival," *IEEE Trans. Signal Process.*, vol. 66, no. 18, pp. 4946–4956, 2018.

[18] H. Jiang, H. Wang, Z. Zheng, and Q. Xu, "Privacy preserved wireless sensor location protocols based on mobile edge computing," *Comput. Secur.*, vol. 84, pp. 393–401, 2019.

[19] S. U. Hussain and F. Koushanfar, "P3: Privacy preserving positioning for smart automotive systems," *ACM Trans. Des. Automat. Electron. Syst.*, vol. 23, no. 6, pp. 1–19, 2018.

[20] T. Shu, Y. Chen, and J. Yang, "Protecting multi-lateral localization privacy in pervasive environments," *IEEE/ACM Trans. Netw.*, vol. 23, no. 5, pp. 1688–1701, Oct. 2015.

[21] A. Alanwar, Y. Shoukry, S. Chakraborty, P. Martin, P. Tabuada, and M. Srivastava, "PrOLoc: Resilient localization with private observers using partial homomorphic encryption," in *Proc. IEEE/ACM 16th Int. Conf. Inf. Process. Sensor Netw.*, 2017, pp. 41–52.

[22] A. Viand, P. Jattke, and A. Hithnawi, "SoK: Fully homomorphic encryption compilers," in *Proc. IEEE Symp. Secur. Privacy*, 2021, pp. 1092–1108.

[23] X. Cao, F. Wang, J. Xu, R. Zhang, and S. Cui, "Joint computation and communication cooperation for energy-efficient mobile edge computing," *IEEE Internet Things J.*, vol. 6, no. 3, pp. 4188–4200, Jun. 2019.

[24] P.-C. Chen, "A non-line-of-sight error mitigation algorithm in location estimation," in *Proc. IEEE Wireless Commun. Netw. Conf.*, 1999, pp. 316–320.

[25] L. Cong and W. Zhuang, "Nonline-of-sight error mitigation in mobile location," *IEEE Trans. Wireless Commun.*, vol. 4, no. 2, pp. 560–573, Mar. 2005.

[26] O. Kanhere and T. S. Rappaport, "Position location for futuristic cellular communications: 5G and beyond," *IEEE Commun. Mag.*, vol. 59, no. 1, pp. 70–75, Jan. 2021.

[27] Y. Zhao, Z. Li, N. Cheng, B. Hao, and X. Shen, "Joint UAV position and power optimization for accurate regional localization in space-air integrated localization network," *IEEE Internet Things J.*, vol. 8, no. 6, pp. 4841–4854, Mar. 2021.

[28] A. Beck, P. Stoica, and J. Li, "Exact and approximate solutions of source localization problems," *IEEE Trans. Signal Process.*, vol. 56, no. 5, pp. 1770–1778, May 2008.

[29] 3GPP, "5G;NG Radio Access Network (NG-RAN);Stage 2 functional specification of User Equipment (UE) positioning in NG-RAN," 3rd Generation Partnership Project (3GPP), version 17.2.0, Technical Specification (TS) 138.305, 2022. [Online]. Available: https://portal.3gpp.org/desktopmodules/Specifications/SpecificationDetails.aspx?specificationId=3310

[30] H. Wang, H. Xu, H. Huang, M. Chen, and S. Chen, "Robust task offloading in dynamic edge computing," *IEEE Trans. Mobile Comput.*, vol. 22, no. 1, pp. 500–514, Jan. 2023.

[31] X. Shi, F. Tong, W.-A. Zhang, and L. Yu, "Resilient privacy-preserving distributed localization against dishonest nodes in Internet of Things," *IEEE Internet Things J.*, vol. 7, no. 9, pp. 9214–9223, Sep. 2020.

[32] E. Dauterman, M. Rathee, R. A. Popa, and I. Stoica, "Waldo: A private time-series database from function secret sharing," in *Proc. IEEE Symp. Secur. Privacy*, 2022, pp. 2450–2468.

[33] A. Patra, T. Schneider, A. Suresh, and H. Yalame, "ABY2.0: Improved mixed-protocol secure two-party computation," in *Proc. USENIX Secur. Symp.*, 2021, pp. 2165–2182.

[34] Y. Zheng et al., "SetRkNN: Efficient and privacy-preserving set reverse kNN query in cloud," *IEEE Trans. Inf. Forensics Secur.*, vol. 18, pp. 888–903, 2023.

[35] S. D. Galbraith, S. W. Gebregiyorgis, and S. Murphy, "Algorithms for the approximate common divisor problem," *LMS J. Comput. Math.*, vol. 19, no. A, pp. 58–72, 2016.

[36] S. Zhang, S. Ray, R. Lu, and Y. Guan, "PPsky: Privacy-preserving skyline queries with secret sharing in eHealthcare," in *Proc. IEEE Glob. Commun. Conf.*, 2022, pp. 5469–5474.

[37] J. Chen, L. Liu, R. Chen, and W. Peng, "SHOSVD: Secure outsourcing of high-order singular value decomposition," in *Proc. 25th Australa. Conf. Inf. Secur. Privacy*, 2020, pp. 309–329.

[38] K. Hamada, R. Kikuchi, D. Ikarashi, K. Chida, and K. Takahashi, "Practically efficient multi-party sorting protocols from comparison sort algorithms," in *Proc. Int. Conf. Inf. Secur. Cryptol.*, 2013, pp. 202–216.

[39] 2023. [Online]. Available: https://github.com/EnderCheng/Localization

[40] G. Wang, J. He, X. Shi, J. Pan, and S. Shen, "Analyzing and evaluating efficient privacy-preserving localization for pervasive computing," *IEEE Internet Things J.*, vol. 5, no. 4, pp. 2993–3007, Aug. 2018.

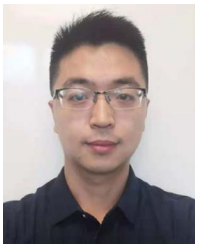
[41] Z. Wang, Y. Xu, Y. Yan, Y. Zhang, Z. Rao, and X. Ouyang, "Privacy-preserving indoor localization based on inner product encryption in a cloud environment," *Knowl.-Based Syst.*, vol. 239, 2022, Art. no. 108005.

[42] R. Nieminen and K. Järvinen, "Practical privacy-preserving indoor localization based on secure two-party computation," *IEEE Trans. Mobile Comput.*, vol. 20, no. 9, pp. 2877–2890, Sep. 2021.

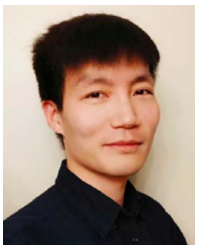
[43] N. Xie, Y. Chen, Z. Li, and D. O. Wu, "Lightweight secure localization approach in wireless sensor networks," *IEEE Trans. Commun.*, vol. 69, no. 10, pp. 6879–6893, Oct. 2021.

Authorized licensed use limited to: University of Waterloo. Downloaded on February 17,2026 at 23:06:10 UTC from IEEE Xplore. Restrictions apply.

Cheng Huang (Member, IEEE) received the PhD degree in electrical and computer engineering from the University of Waterloo, ON, Canada, in 2020. He is currently an associate professor with the School of Computer Science at Fudan University. Before joining Fudan University, he was a research fellow with the Department of Electrical and Computer Engineering at the University of Waterloo from 2020 to 2023. His research interests lie in the areas of security and privacy in vehicular networks, data security, and secure computation. He has published more than 60 papers in prestigious journals and conferences, including *IEEE Transactions on Dependable and Secure Computing*, *IEEE Journal on Selected Areas in Communications*, *IEEE Transactions on Vehicular Technology*, and *IEEE Transactions on Industrial Informatics*, and has received Best Paper Awards from ICCC'15, ICC'18, GLOBECOM'22, and ICCC'23. He serves as the associate editor for Peer-to-Peer Networking and Applications (Springer), the Symposium Chair of IEEE GLOBECOM'24, and has served as the publicity chair of ICA3PP'22, PST'23, SustainCom'23, and as a TPC member of many international conferences.



Dongxiao Liu (Member, IEEE) received the PhD degree from the Department of Electrical and Computer Engineering, University of Waterloo, Canada in 2020. He was a postdoctoral research fellow with the Department of Electrical and Computer Engineering, University of Waterloo from 2020 to 2023. He is now with the School of Computer Science and Engineering, University of Electronic Science and Technology of China. His research interests include security and privacy in communication networks and blockchain.



Anjia Yang (Member, IEEE) received the PhD degree from the department of Computer Science from the City University of Hong Kong in 2015. He held a postdoctoral position with the City University of Hong Kong from 2015 to 2016, and in Jinan University from 2016 to 2019, respectively. From 2018 to 2019, he was a visiting scholar in BBCR group in University of Waterloo. He is currently a professor in Jinan University, Guangzhou. His research interests include security and privacy in Internet of Things, vehicular networks, blockchain and cloud computing, etc. He has published more than 60 international papers including journals and conferences, such as *IEEE Transactions on Information Forensics and Security*, *IEEE Transactions on Dependable and Secure Computing*, *IEEE Transactions on Mobile Computing*, *IEEE Transactions on Parallel and Distributed Systems*, *IEEE Transactions on Services Computing*, *IEEE Transactions on Intelligent Transportation Systems*, NDSS, ASIACRYPT, and ESORICS. He served as PC members or organizers for more than 30 international conferences. He also serves as editors for journals such as *Security and Communication Networks*, *Symmetry*, *Blockchain*.

Rongxing Lu (Fellow, IEEE) received the PhD degree from the Department of Electrical and Computer Engineering, University of Waterloo, Canada, in 2012. He is the Acting director of Canadian Institute for Cybersecurity (CIC), a Mastercard IoT Research chair, and a full professor with the Faculty of Computer Science (FCS), University of New Brunswick (UNB), Canada. Before that, he worked as an assistant professor with the School of Electrical and Electronic Engineering, Nanyang Technological University (NTU), Singapore from 2013 to 2016. He worked as a postdoctoral fellow with the University of Waterloo from 2012 to 2013. He was awarded the most prestigious "Governor General's Gold Medal", from the Department of Electrical & Computer Engineering, University of Waterloo, Canada, in 2012; and won the 8th IEEE Communications Society (ComSoc) Asia Pacific (AP) Outstanding Young Researcher Award, in 2013. His research interests include applied cryptography, privacy enhancing technologies, and IoT-Big Data security and privacy. He has published extensively in his areas of expertise with H-index 86 and citations 33,100+ from Google Scholar as of January 2024, and was the recipient of 10 best (student) paper awards from some reputable journals and conferences. He served/ serves as the Chair of 2022-2023 IEEE ComSoc CIS-TC (Communications and Information Security Technical Committee), and the founding Co-chair of IEEE TEMS Blockchain and Distributed Ledgers Technologies Technical Committee (BDLT-TC). He is the Winner of 2016-17 Excellence in Teaching Award, FCS, UNB.

Xuemin (Sherman) Shen (Fellow, IEEE) received the PhD degree in electrical engineering from Rutgers University, New Brunswick, NJ, USA, in 1990. He is a University professor with the Department of Electrical and Computer Engineering, University of Waterloo, Canada. His research focuses on network resource management, wireless network security, Internet of Things, 5G and beyond, and vehicular networks. He is a registered professional engineer of Ontario, Canada, an Engineering Institute of Canada fellow, a Canadian Academy of Engineering fellow, a Royal Society of Canada fellow, a Chinese Academy of Engineering foreign member, and a distinguished lecturer of the IEEE Vehicular Technology Society and Communications Society. He received "West Lake Friendship Award" from Zhejiang Province in 2023, President's Excellence in Research from the University of Waterloo in 2022, the Canadian Award for Telecommunications Research from the Canadian Society of Information Theory (CSIT) in 2021, the R.A. Fessenden Award in 2019 from IEEE, Canada, Award of Merit from the Federation of Chinese Canadian Professionals (Ontario) in 2019, James Evans Avant Garde Award in 2018 from the IEEE Vehicular Technology Society, Joseph LoCicero Award in 2015 and Education Award in 2017 from the IEEE Communications Society (ComSoc), and Technical Recognition Award from Wireless Communications Technical Committee (2019) and AHSN Technical Committee (2013). He has also received the Excellent Graduate Supervision Award in 2006 from the University of Waterloo and the Premier's Research Excellence Award (PREA) in 2003 from the Province of Ontario, Canada. He serves/served as the General Chair for the 6G Global Conference'23, and ACM Mobihoc'15, Technical Program Committee Chair/Co-Chair for IEEE Globecom'24, 16 and 07, IEEE Infocom'14, IEEE VTC'10 Fall, and the Chair for the IEEE ComSoc Technical Committee on Wireless Communications. He is the president of the IEEE ComSoc. He was the vice president for Technical & Educational Activities, vice president for Publications, member-at-Large on the Board of Governors, chair of the Distinguished Lecturer Selection Committee, and Member of IEEE fellow Selection Committee of the ComSoc. He served as the editor-in-chief of the *IEEE Internet of Things Journal*, *IEEE Network*, and *IET Communications*.

Authorized licensed use limited to: University of Waterloo. Downloaded on February 17,2026 at 23:06:10 UTC from IEEE Xplore. Restrictions apply.