Received: 21 September 2022 | Revised: 23 August 2023 | Accepted: 15 September 2023

DOI: 10.33012/navi.625

ORIGINAL ARTICLE

Privacy-Preserving Cooperative GNSS Positioning

Guillermo Hernandez | **Gerald LaMountain** | **Pau Closas**

Authors are with the Department of Electrical and Computer Engineering at Northeastern University, Boston, MA, USA

Correspondence
Guillermo Hernandez, Department of Electrical and Computer Engineering at Northeastern University, Boston, MA, USA.
Email: hernandez.g@northeastern.edu

Funding Information
National Science Foundation, Grant/Award Number: ECCS-1845833 and CCF-2326559

Abstract

The issue of user privacy in the context of collaborative positioning is addressed in this work, wherein information is passed between and processed by multiple cooperative agents, with the goal of achieving high levels of positioning accuracy. In particular, this paper discusses three privacy-preserving schemes in the context of differential global navigation satellite system (GNSS)-based and GNSS-based cooperative positioning methods. The discussed architectures provide the same positioning results, while yielding different levels of privacy to the cooperative users. These architectures also involve increased complexity as privacy grows and as non-encrypted, encrypted, and homomorphically encrypted solutions are implemented. The latter scheme is the most computationally demanding; however, it provides the highest level of privacy by employing homomorphic encryption, whereby addition and multiplication operations may be performed on encrypted data to produce encrypted outputs, without revealing information about the collaborative agent's location. The proposed privacy-preserving cooperative position schemes are shown to provide the same results as their non-privacy-preserving counterparts, while providing privacy guarantees. Based on this analysis, some of the proposed solutions can be considered for real-time applications, while homomorphic encryption is a valid solution for latency-tolerant applications. Advances in computing power will increase their overall usability in the near future.

Keywords

cooperative positioning, differential GNSS, GNSS, homomorphic encryption, privacy-preserving

1 | INTRODUCTION

Among the ever-expanding toolbox of methods for providing positioning in modern technology, few are as continually prevalent as global navigation satellite systems (GNSSs) (Borre et al., 2007; Hofmann-Wellenhof et al., 2007; Kaplan, 2006; Morton et al., 2021; Pany, 2010). As the GNSS research community continues to expand and investigate innovative ideas to increase the availability, precision, and reliability of these systems (Amin et al., 2016; Closas et al., 2017; Dardari et al., 2015 2011; Kassas et al., 2019), one technique that has been gaining traction is "collaborative" positioning (Garello et al., 2012; Huang et al., 2015 2014; Minetto et al., 2017).

In collaborative positioning, information is passed between and processed by multiple collaborators, with the goal of minimizing interference and achieving high levels of positioning accuracy for each or some of the collaborators. The application of obtaining highly accurate location values in an environment that involves multiple collaborators has expanded in conjunction with emerging technologies such as smart cities and intelligent transportation systems technology (Alsamhi et al., 2019; Laoudias et al., 2018; Tahir et al., 2018; Williams et al., 2022). Additionally, technologies embedded with GNSS receivers, such as smartphone and vehicular communication systems, have expanded by incorporating collaborative positioning frameworks in order to enhance the accuracy of positioning performance by mitigating the errors observed for standalone GNSS, for instance, in the context of cooperative vehicular networks (Alam & Dempster, 2013), for COVID-19 contact tracing purposes (Minetto et al., 2022), to exploit a massive number of users for enhanced differentiation schemes (Calatrava et al., 2023), to exploit neighboring user positions (Hernandez et al., 2023), or to distributedly learn jammer classifiers (Wu et al., 2023).

Although these techniques may be desirable for their increase in precision, it may not always be advisable to expose information that might be used to identify a user's location to collaborators, as this information may become vulnerable to wireless and cyber attacks. Previous studies have researched the vulnerability of such systems that depend on user position and location information. In aviation systems, instrumental landing systems that depend on precision approach systems may become vulnerable to wireless attacks that may change the instructions provided by the system to the pilot (Ranganathan & Capkun, 2017; Sathaye et al., 2019). As advances continue, it appears that everyday technology is increasing its dependency on proximity-based and location-based technology to make everyday tasks easier; however, this technology may become defenseless to certain attacks (Ranganathan & Capkun, 2017). In certain situations, these attacks may access the data layer of the system and modify the distance on which these systems depend. Thus, researchers are interested in the task of keeping a user's location private while performing the computations necessary to determine the user's position at a remote location from GNSS observables. By making a user's location private, a collaborative positioning framework may prevent an outside adversary, who is not part of the framework, from infiltrating a framework to collect user information. Moreover, this approach will account for a honest but curious user, or an untrusted user, who partakes in such a framework with the objective of obtaining user information. By addressing these privacy concerns, users will be encouraged to partake in a collaborative positioning framework, similar to those mentioned above.

Multiple approaches have been proposed for addressing privacy concerns in location-based applications, as well as different perspectives on where these privacy concerns may occur (Chen et al., 2017). For instance, Lohan et al. (2022) proposed a perturbation method that depends on random noise that is suitable for proximity-detection-based services and applications that rely on the relative distance between two receivers. Before broadcasting an approximated location to the server, a user may control the level of perturbed noise to add to its location values, while satisfying the noise threshold criteria. With this approximated location, the server can identify other users near this location. It was found that the privacy of the proposed mechanism depends on the level of accuracy; a more accurate location requires less inserted noise. Additionally, the proposed mechanism depends on external services and applications. Holcer et al. (2020) identified three locations in which location privacy-preserving mechanisms are used: on the device, in transmitted data, and at on-server locations. A common approach for addressing privacy

concerns for on-server locations is ϵ-differential privacy, which maintains a user's information within a statistical database (Feng et al., 2019). While the ϵ-differential privacy method provides user privacy, a user's position may still be exposed within the application server.

In this work, we evaluate three different approaches to address the task of preserving user information. The first method is not based on encrypted communications, and it is shown to limit the exposure of a user's information. With this method, the only possible information leakage entails the user's position uncertainty, but not the user's position information. This position uncertainty could be accessed by a user's collaborative partner and an eavesdropping adversarial user. The second method utilizes regular encryption methodologies in order to maintain the privacy of information transmitted between users within the framework. We discuss how this approach prevents a third party from eavesdropping on the communication flow to learn about the cooperative users' position uncertainty, as well as their positions. The encrypted data possess properties that allow sensitive information to be concealed, and because of these properties, the encrypted data can be broadcast to one or more collaborators without compromising the message's content. This effectively prevents an outside observer from obtaining any valuable information from an encrypted message. Additionally, with an encryption scheme method, a user does not depend on an external server, such as a localization server that can access a user's measurements, to compute its position. While this approach maintains the message's integrity against external adversarial users, vulnerability arises when the recipient of the encrypted data decrypts it, subsequently exposing their collaborative counterpart's position uncertainty information. Therefore, the third approach exploits the use of "homomorphic" encryption schemes for private cooperative positioning (CoPo). This approach retains the same benefits as the second approach, the regular encryption approach, while providing the additional feature that cooperative users cannot learn the position uncertainty or position of their peers. By using homomorphic encryption, certain elementary mathematical operations may be performed on encrypted data, or ciphertext, resulting in an encrypted output which, when decrypted, is identical to that which would have been computed had encryption not been employed. The benefit of this property is that the manipulations required for collaboration can be performed in a way that does not expose the details of the operation, either at the input or the output, to the user performing the manipulations.

In our preliminary work (Hernandez et al., 2020), homomorphic encryption was proposed to leverage privacy in collaborative and differential GNSS (DGNSS) schemes. The current article extends that previous work by broadening the discussion toward different approaches of privacy-preserving methods, including a non-encryption scheme and two encryption-based schemes. In parallel, the discussion is also expanded to the applicability of a weighted least-squares (WLS) algorithm, while considering the least-squares (LS) algorithm as a particular case. Additionally, the complexity aspects of each approach are discussed, providing a deeper experimental discussion based on simulated GNSS observable values obtained by an "open-sky" environment, and each approach is also analyzed using real-world data. In summary, the objective of this article is to demonstrate how each of these different approaches can provide a relevant satellite navigation scheme for situations in which private data (i.e., user position) must be shared and to expand the discussion on how these approaches can serve as a protocol for achieving such privacy in single-difference positioning.

This article demonstrates that the approaches employed in satellite navigation can effectively address privacy-preserving concerns within single-difference

positioning, as well as potentially other CoPo schemes. The CoPo algorithm may utilize these privacy-preserving schemes to generate estimates for position, velocity, and time (PVT) values. These estimates are derived from information exchanged with neighboring users, resembling code-phase DGNSS schemes, while also incorporating privacy-preserving features. These methodologies leverage a single pseudorange difference that cancels the effects of certain impairments (i.e., tropospheric delay and ionospheric delay) between two receivers (Gogoi et al., 2019). The two receivers, or users, compute the position of the other user while using one of these privacy-preserving approaches to perform the computations securely. Afterward, they return the results to the respective user so that each user can determine its final estimated position.

As part of the simulation process, the required computations are performed locally at each receiver, leveraging the data of others. Thus, with respect to user privacy, the variables involved in the WLS (and LS) algorithm should remain private and be sent to the users responsible for computing the PVT solution. When the different methodologies that provide privacy are used in a WLS problem (in this case, the PVT computation), the results are expected to be the same as those for a WLS problem with a non-privacy-preserving scheme in place. Within the simulation process, consideration is given to what was briefly described, and the results will establish a concrete understanding that utilizing a secure protocol has little to no effect in terms of performance degradation while adding an important layer of privacy to the user accessing the service. Parallel to this idea, the problem with the non-privacy-preserving scheme will serve as a base for comparing the accuracy of the results to the cases in which a privacy-preserving scheme is implemented.

The remainder of this paper is organized into three main sections. Section 2 details a CoPo scheme and its implementation of the WLS and LS algorithms. Section 3 provides some technical background on homomorphic encryption, which is required in order to implement the private CoPo scheme. In Section 4, the three approaches for preserving privacy between collaborating peers are presented. Subsequently, in Section 5, the results of simulated and real-world data are discussed to validate the three privacy-preserving schemes, and conclusions are drawn in Section 6.

2 | COPO METHODOLOGY

A positioning scheme is described in which two arbitrarily close receivers exchange code-phase GNSS observables such that they can remove atmospheric errors by combining those measurements. Under such a scheme, pseudoranges measured by two receivers (denoted by n and m indices) are subtracted, in the vein of standard DGNSS schemes, with the particularity that none of those receivers have accurate knowledge of their own position and that each receiver benefits from the process (i.e., no base station is present). The remainder of this paper leverages this CoPo scheme to propose an approach that preserves privacy among these collaborating users; that is, the CoPo scheme can be implemented without the n-th user being able to infer the position of the m-th user, and vice versa.

The remainder of this section details the formulation of the aforementioned CoPo scheme to later describe how it can be practically implemented in three alternative schemes. Namely, a non-encryption, a regular encryption, and a homomorphic encryption implementation can be leveraged to preserve privacy among participating users, each providing increasing levels of privacy to the participating users as discussed in Section 4.

2.1 | DGNSS as a CoPo Scheme

A GNSS receiver processes signals to compute the pseudorange and carrier phase measurements for L visible satellites. Let us assume that two nearby receivers (n and m) are able to receive their respective navigation signals from the same set of satellites, as shown in Figure 1, to compute the pseudorange observable. As an example, the n-th receiver is modeled as follows:

$$\rho_i^{(n)} = \| \boldsymbol{p}^{(n)} - \boldsymbol{p}_i \| + c(\delta t^{(n)} - \delta t_i) + T_i + I_i + \epsilon_i^{(n)} \tag{1}$$

such that $\boldsymbol{p}^{(n)}$ denotes the position vector of the n-th receiver; $\boldsymbol{p}_i$ the position of the i-th satellite with $i = \{1, \ldots, L\}$; $\delta t^{(n)}$ and δt_i are the clock offsets of the receiver (unknown) and the satellite (known), respectively; c is the speed of light; T_i and I_i are the delay errors due to troposphere and ionosphere effects, respectively; and $\epsilon_i^{(n)} \sim \mathcal{N}(0, \sigma_{n,i}^2)$ is a random variable representing the pseudorange estimation error as well as other unmodeled terms. The joint covariance matrix is then $\mathbf{R}^{(n)} = \text{diag}(\sigma_{n,1}^2, \ldots, \sigma_{n,L}^2)$. In vector form, the L pseudoranges can be gathered as follows:

$$\boldsymbol{\rho}^{(n)} = (\rho_1^{(n)}, \cdots, \rho_L^{(n)})^\top \tag{2}$$

By computing the difference between the pseudoranges observed by two neighboring receivers, a new observable can be obtained that is free of the common troposphere- and ionosphere-associated delays:

$$\begin{aligned} \Delta\rho_i^{(n,m)} &= \rho_i^{(n)} - \rho_i^{(m)} \\ &= \| \mathbf{p}^{(n)} - \mathbf{p}_i \| - \| \mathbf{p}^{(m)} - \mathbf{p}_i \| + c(\delta t^{(n)} - \delta t^{(m)}) + \eta_i^{(n,m)} \end{aligned} \tag{3}$$

Equation (3) does not have the contributions of T_i and I_i. The random term is $\eta_i^{(n,m)} = \epsilon_i^{(n)} - \epsilon_i^{(m)} \sim \mathcal{N}(0, \sigma_{n,i}^2 + \sigma_{m,i}^2)$, whose variance increases as a consequence of the process. This parameter is represented in vector form:

$$\Delta\boldsymbol{\rho}^{(n,m)} = (\Delta\rho_1^{(n,m)}, \cdots, \Delta\rho_L^{(n,m)})^\top \tag{4}$$

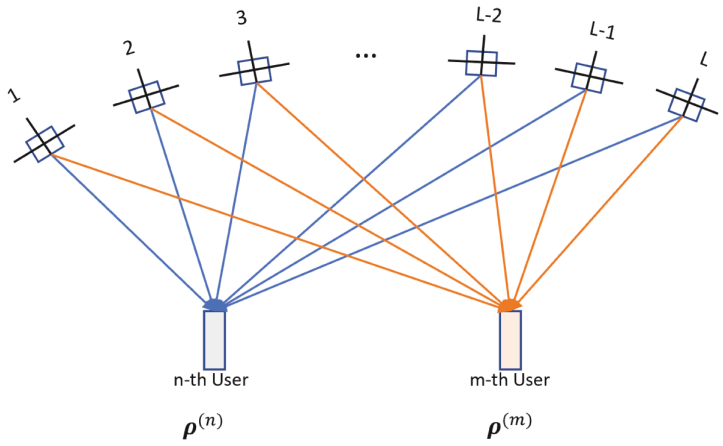

FIGURE 1 In differential positioning, all receivers observe navigation signals from the same set of satellites.

In general, the combined pseudoranges are modeled as a nonlinear function of the unknown position and clock offset of the receivers:

$$\Delta\boldsymbol{\rho}^{(n,m)} = \mathbf{h}(\boldsymbol{x}^{(n)}, \boldsymbol{x}^{(m)}) + \boldsymbol{\eta} \tag{5}$$

where $\boldsymbol{\eta} = (\eta_1^{(n,m)}, \cdots, \eta_L^{(n,m)})^\top \sim \mathcal{N}(\mathbf{0}, \mathbf{R}^{(n)} + \mathbf{R}^{(m)})$; $\boldsymbol{x}^{(n)}$ is a vector gathering $\boldsymbol{p}^{(n)}$ and $c\delta t^{(n)}$; and $\mathbf{h} : \mathbb{R}^4 \times \mathbb{R}^4 \mapsto \mathbb{R}^L$ is a mapping from position and time unknowns to combine pseudoranges, which makes this a difficult problem to solve in general. However, the mapping is known and given by the following[1]:

$$[\mathbf{h}(\boldsymbol{x}^{(n)}, \boldsymbol{x}^{(m)})]_i = \| \mathbf{p}^{(n)} - \mathbf{p}_i \| - \| \mathbf{p}^{(m)} - \mathbf{p}_i \| + c(\delta t^{(n)} - \delta t^{(m)}) \tag{6}$$

One can take a first-order Taylor expansion to linearize the problem at some arbitrary points $\mu_0^{(n)}$ and $\mu_0^{(m)}$:

$$\mathbf{h}(\boldsymbol{x}^{(n)}, \boldsymbol{x}^{(m)}) \approx \mathbf{h}(\mu_0^{(n)}, \mu_0^{(m)}) + \nabla_{\boldsymbol{x}} \mathbf{h}(\mu_0^{(n)}, \mu_0^{(m)})(\boldsymbol{x} - \mu_0) \tag{7}$$

Here, $\boldsymbol{x}$ gathers both $\boldsymbol{x}^{(n)}$ and $\boldsymbol{x}^{(m)}$, and $\boldsymbol{H} \triangleq \nabla_{\boldsymbol{x}} \mathbf{h}(\cdot)$ denotes the derivative of $\mathbf{h}(\cdot)$ with respect to $\boldsymbol{x}$, which can be computed as $\boldsymbol{H} = [\boldsymbol{H}^{(n)}, -\boldsymbol{H}^{(m)}]$ where we have the following:

$$\boldsymbol{H}^{(n)} = \begin{pmatrix} -\boldsymbol{u}_1^\top(\boldsymbol{x}^{(n)}) & 1 \\ \vdots & \vdots \\ -\boldsymbol{u}_L^\top(\boldsymbol{x}^{(n)}) & 1 \end{pmatrix} \tag{8}$$

Once the problem in Equation (5) is linearized, it can be easily solved analytically through an iterative LS procedure. To do so, the terms are rearranged as follows:

$$\boldsymbol{y}^{(n,m)} = \Delta\boldsymbol{\rho}^{(n,m)} + \boldsymbol{H}\mu_0 - \mathbf{h}(\mu_0^{(n)}, \mu_0^{(m)}) \tag{9}$$

$$\left[\mathbf{h}(\mu_0^{(n)}, \mu_0^{(m)})\right]_i = \underbrace{\| \mathbf{p}_0^{(n)} - \mathbf{p}_i \| + c\delta t_0^{(n)}}_{\left[\mathbf{h}(\mu_0^{(n)})\right]_i} - \underbrace{\| \mathbf{p}_0^{(m)} - \mathbf{p}_i \| - c\delta t_0^{(m)}}_{\left[\mathbf{h}(\mu_0^{(m)})\right]_i} \tag{10}$$

$$\boldsymbol{y}^{(n,m)} \simeq \boldsymbol{H} \begin{bmatrix} \boldsymbol{p}^{(n)} \\ c\delta t^{(n)} \\ \boldsymbol{p}^{(m)} \\ c\delta t^{(m)} \end{bmatrix} + \boldsymbol{\eta} = \boldsymbol{H}^{(n)} \begin{bmatrix} \boldsymbol{p}^{(n)} \\ c\delta t^{(n)} \end{bmatrix} - \boldsymbol{H}^{(m)} \begin{bmatrix} \boldsymbol{p}^{(m)} \\ c\delta t^{(m)} \end{bmatrix} + \boldsymbol{\eta} \tag{11}$$

where $\boldsymbol{H} = [\boldsymbol{H}^{(n)}, -\boldsymbol{H}^{(m)}]$ can be split into the terms associated with each receiver.

Solving for $\hat{\boldsymbol{x}}^{(m)}$ in Equation (11) as a WLS problem and assuming that the n-th receiver has an estimate of $\hat{\boldsymbol{x}}^{(n)}$, the WLS estimate for the m-th receiver can be readily computed as follows:

$$\hat{\boldsymbol{x}}^{(m)} = (\boldsymbol{H}^{(m)\top} \boldsymbol{R}^{-1} \boldsymbol{H}^{(m)})^{-1} \boldsymbol{H}^{(m)\top} \boldsymbol{R}^{-1} (\boldsymbol{H}^{(n)} \hat{\boldsymbol{x}}^{(n)} - \boldsymbol{y}^{(n,m)}) \tag{12}$$

$$= \mu_0^{(m)} - (\boldsymbol{H}^{(m)\top} \boldsymbol{R}^{-1} \boldsymbol{H}^{(m)})^{-1} \boldsymbol{H}^{(m)\top} \boldsymbol{R}^{-1} \cdot \mathbf{T}^{(n)} \tag{13}$$

[1] $[a]_i$ denotes the i-th element in vector $\mathbf{a}$. $[\mathbf{A}]_{i,j}$ denotes the $\{i, j\}$-th element in matrix $\mathbf{A}$.

where $\mathbf{T}^{(n)} = (\boldsymbol{y}^{(n,m)} - \boldsymbol{H}^{(n)}\hat{\boldsymbol{x}}^{(n)} + \boldsymbol{H}^{(m)}\boldsymbol{\mu}_0^{(m)})$. Similarly, we have the following:

$$\hat{\boldsymbol{x}}^{(n)} = (\boldsymbol{H}^{(n)\top}\boldsymbol{R}^{-1}\boldsymbol{H}^{(n)})^{-1}\boldsymbol{H}^{(n)\top}\boldsymbol{R}^{-1}(\boldsymbol{y}^{(n,m)} + \boldsymbol{H}^{(m)}\hat{\boldsymbol{x}}^{(m)}) \tag{14}$$

$$= \boldsymbol{\mu}_0^{(n)} + (\boldsymbol{H}^{(n)\top}\boldsymbol{R}^{-1}\boldsymbol{H}^{(n)})^{-1}\boldsymbol{H}^{(n)\top}\boldsymbol{R}^{-1} \cdot \mathbf{T}^{(m)} \tag{15}$$

where $\mathbf{T}^{(m)} = (\boldsymbol{y}^{(n,m)} + \boldsymbol{H}^{(m)}\hat{\boldsymbol{x}}^{(m)} - \boldsymbol{H}^{(n)}\boldsymbol{\mu}_0^{(n)})$ for the other unknown vector. Note that solving this WLS problem requires knowledge of the pseudorange variances of both receivers because $\mathbf{R} = \mathbf{R}^{(n)} + \mathbf{R}^{(m)}$.

In the case of the LS problem, the measurements are assumed to be independent and identically distributed such that the covariance matrix of the measurements is diagonal, $\boldsymbol{R} = \sigma^2\boldsymbol{I}$. Thus, we obtain the following:

$$\begin{aligned}\hat{\boldsymbol{x}}^{(n)} &= (\boldsymbol{H}^{(n)\top}\boldsymbol{H}^{(n)})^{-1}\boldsymbol{H}^{(n)\top}(\boldsymbol{y}^{(n,m)} + \boldsymbol{H}^{(m)}\hat{\boldsymbol{x}}^{(m)}) \\ &= \boldsymbol{\mu}_0^{(n)} + (\boldsymbol{H}^{(n)\top}\boldsymbol{H}^{(n)})^{-1}\boldsymbol{H}^{(n)\top} \cdot \mathbf{T}^{(m)}\end{aligned} \tag{16}$$

A similar approach is considered for the estimated value of the n-th receiver, $\hat{\mathbf{x}}^{(n)}$. With this approach, it is possible for receivers to estimate their PVT values. For the situation in which each receiver is substituted for a user and each user shares their own measurements as described by the procedure, this process does not account for user privacy. Therefore, we present three possible schemes implementing a differential positioning scheme that addresses this privacy concern at different levels in Section 4, namely, a non-encrypted scheme, a regular encryption scheme, and a scheme using homomorphic encryption.

3 | HOMOMORPHIC ENCRYPTION RUDIMENTS

Homomorphic encryption schemes fall broadly into three categories: partially homomorphic encryption (PHE), somewhat homomorphic encryption (SWHE), and fully homomorphic encryption (FHE). The distinctions between each of these categories are based on the type and number of operations that may be performed on a piece of encrypted data before it is unrecoverable via decryption. In a PHE scheme, encryption is limited to only one operation performed over ciphertexts (encrypted data): addition or multiplication. In contrast, an SWHE scheme limits the number of operations performed, but allows both addition and multiplication to be performed over ciphertexts. Finally, in many ways combining the advantages of both PHE and SWHE, FHE schemes can be used to perform an unlimited number of both addition and multiplication operations over ciphertexts (Acar et al., 2018)

Research on FHE schemes has expanded substantially since the first functional scheme was introduced by Gentry (Gentry, 2009; Gentry et al., 2013; van Dijk et al., 2010). Since then, many have used the Gentry blueprint to develop newer FHE schemes based on the learning-with-errors (LWE) problem. In general, FHE schemes are computationally expensive and complex during a multiplication operation, and a few researchers have attempted to minimize these common issues. Within the proposed privacy-preserving CoPo approach, the Brakerski, Fan, and Vercauteren (BFV) scheme is taken into consideration (Brakerski, 2012; Fan & Vercauteren, 2012). The BFV scheme used herein describes the process for converting an SWHE scheme into a FHE scheme.

An encryption scheme generally consists of three main algorithms; the key generation algorithm $\mathsf{KeyGen}(\cdot)$, an encryption algorithm $\mathsf{Enc}(\cdot)$, and a decryption

algorithm Dec(·). To gain a better understanding of these algorithms, it is beneficial to examine the case in which the n-th user seeks to communicate with the m-th user in a private manner. The n-th user generates a set of public and private keys through the use of the KeyGen(·) algorithm. The n-th user must distribute its public key to the m-th user, but never distributes its private key, thus solely maintaining access to its private key. With the n-th user's public key, the m-th user may now conceal its message and send this hidden message to the n-th user. The process of using a public key to hide a message is considered the encryption process, which is accomplished through the encryption algorithm Enc(·). Therefore, the encryption algorithm takes a message as input and produces a ciphertext (the encrypted message) as output. When the n-th user receives the hidden message from the m-th user, the n-th user also receives the ciphertext. With the ciphertext and its private key, the n-th user is able to retrieve the message within the ciphertext. The process of retrieving the message within a ciphertext with the private key is the decryption process, which is performed during the decryption algorithm, Dec(·).

In addition to concealing a message, an FHE system will support one or more mathematical operations. In such a system, the operator can be applied to the encrypted form of a set of arguments, and the desired result is attained by decryption. Specifically, denoting encryption by Enc(·), decryption by Dec(·), and an operator by $f(\cdot,\cdot)$, the following holds for homomorphic encryption:

$$\mathsf{Dec}(f(\mathsf{Enc}(m_1),\mathsf{Enc}(m_2))) = f(m_1, m_2) \tag{17}$$

for two messages m_1 and m_2 (van Dijk et al., 2010). That is, the function can be applied to encrypted data, which are then decrypted, and yield the same result as when privacy is not considered. The types of functions supported (typically multiplication and addition) depend on the considered cryptosystem (van Dijk et al., 2010).

The BFV scheme follows this approach. This scheme contains the three algorithms, KeyGen(·), Enc(·), and Dec(·), and it supports the operations (multiplication and addition) used for the CoPo scheme. The BFV scheme uses two sets of rings, one for the plaintext and the other for the ciphertext. The polynomial ring $R = \mathbb{Z}[x]/(f(x))$, where $f(x) \in \mathbb{Z}[x]$ is a monic irreducible polynomial of degree d. A popular choice for $f(x)$ is $x^d + 1$, a cyclotomic, where $d = 2^n$. The ring R contains coefficients from the set of integers $(-q/2, q/2]$, denoted as $\mathbb{Z}_q$, where $q > 1$. Therefore, R_q denotes the set of polynomials in R with coefficients in $\mathbb{Z}_q$. From this, we have both the plaintext polynomial space and the ciphertext polynomial space, denoted as R_t and R_q, respectively.

In the encryption process, a plaintext polynomial serves as the input, and a new polynomial is generated, incorporating random coefficient values. These two polynomials are then added together, resulting in a modified version of the original polynomial. The generated polynomial, considered as noise, plays a critical role in ensuring the required security and performing various operations. The plaintext polynomial is then mapped to the ciphertext space to attain a ciphertext.

During the decryption process, the ciphertext is reverted back to the plaintext polynomial via the private key, and the noise is simultaneously removed. It is possible to obtain an incorrect decryption when the noise observed during the encryption process exceeds a certain noise threshold. The noise increases when operations are performed on the ciphertext; thus, if the noise remains under the noise threshold, the original plaintext polynomial is successfully obtained. However, if the noise exceeds the threshold, the plaintext polynomial will differ from the original plaintext polynomial, leading to an incorrect decryption.

The BFV scheme supports ciphertext addition and multiplication operations. When two ciphertexts are added together, the coefficients of the underlying polynomials are added in a coefficient-wise manner. As a consequence, the noises of each polynomial are combined, resulting in an increase in the overall noise level. This noise is measured in terms of bit size. With the addition operation, the noise will increase by approximately one bit. During the multiplication operation, the polynomials are multiplied by a polynomial multiplication technique. Similarly, the amount of noise increases at a rapid rate, which will cause the amount of noise to increase by several bits.

Certain parameters influence the noise threshold level and, in parallel, the security of the system. The main parameters include the polynomial degree, the modulus parameter for the polynomials, and the security parameter. The polynomial degree, denoted as n, has a crucial role in determining both the computational cost and the size of the polynomials for the plaintext and ciphertext. Increasing the polynomial degree results in larger polynomials, enabling more complex computations but also leading to a higher computational cost.

The plaintext modulus parameter, denoted as t, provides the range of the plaintext coefficients. As t increases, the range of coefficients expands. Similarly, the ciphertext modulus parameter, denoted as q, affects the range of coefficients in the ciphertext. These two parameters significantly impact the noise level and noise threshold.

When t increases, the result from a computed operation will be large. Consequently, the noise level, or noise growth, increases at a higher rate compared with when t is smaller. In contrast, the q parameter impacts the noise threshold. Increasing q will increase the ciphertext coefficient range, therefore allowing a higher acceptable amount of noise, thus increasing the noise threshold. Furthermore, when considering the value of q, the bit size should be similar to the polynomial degree, n. If n is larger than q, there will be an unnecessary increase in the computational complexity. Conversely, if n is much less than q, then the level of computation capability becomes limited.

The security level, denoted as λ, impacts the scheme security by determining the noise introduced to the polynomial during the encryption process. As the security level increases, the noise magnitude becomes larger for the generated random polynomial. This noise increase also increases the computational capacity and the resources required for an adversary to break the scheme. However, as the noise magnitude increases, q must increase in order to raise the noise threshold. It is important to emphasize that $\lambda, n,$ and q are not independent. When considering security, there is a trade-off between n and q; for more security, n is increased but q must decrease. The aspect of computational complexity must also be considered. To reduce the complexity, both n and q must be small. Lastly, the accuracy depends on q: as this value increases, the outcome becomes more accurate.

Additional technical details can be found in Appendix A, which further discusses the relevant parameters and addition/multiplication operations within the homomorphic encryption frameworks.

4 | PRIVACY-PRESERVING COPO METHODOLOGY

This section details three implementations of the CoPo scheme discussed in Section 2. Namely, we implement 1) a scheme that leverages a convenient rearrangement of Equation (14) and that does not employ encryption techniques, 2) a scheme that adds encrypted communications to the former, and 3) a scheme

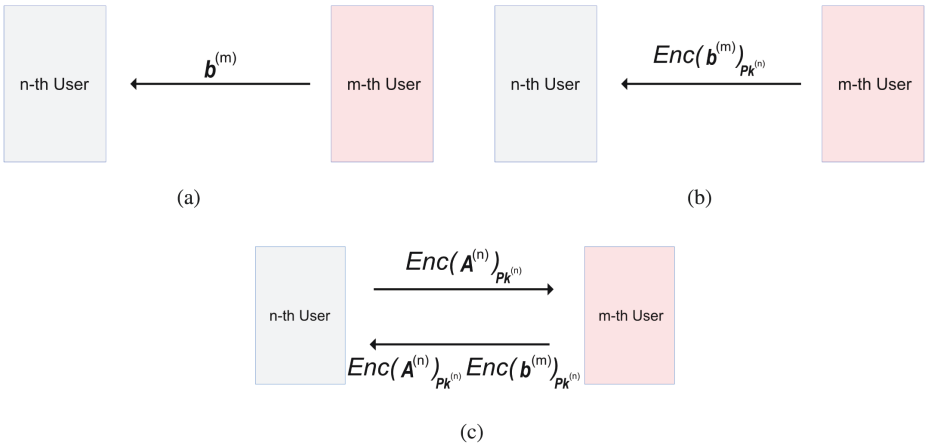

FIGURE 2 Three different schemes that maintain user privacy
(a) In the non-encryption scheme, the m-th user sends $\mathbf{b}^{(m)}$ to the n-th user. (b) In the regular encryption scheme, the n-th user shares its public key $\mathbf{Pk}^{(n)}$ with the m-th user, and the m-th user sends $\mathbf{b}^{(m)}$ encrypted. (c) In the homomorphic encryption scheme, the the n-th user shares its public key $\mathbf{Pk}^{(n)}$ with the m-th user and also sends $\boldsymbol{A}^{(n)}$ encrypted. The m-th user then proceeds to encrypt its own data using $\mathbf{Pk}^{(n)}$, computes the corresponding terms in Equation (24), and sends the encrypted matrix/vector product to the n-th user.

that implements a protocol in which homomorphic encryption is used to provide an additional degree of privacy. A simple illustration of each scheme is shown in Figure 2. This section also discusses the privacy features of each approach as well as the attack models that they can each handle. In collaborative processes, each user computes a portion of the other's position in a private manner and communicates it back to its peer in an iterative process that mitigates the effect of atmospheric errors.

4.1 | Non-Encryption Approach

Here, a "privacy-aware" modification to the CoPo scheme previously described in Section 2 is proposed, in which each user can obtain an enhanced PVT solution after combining the pseudorange by solving Equations (12) and (14). The first modification assumes that users have the capability to perform the required computations and are able to partake in the collaborating component while users can enjoy the benefits afforded by CoPo without exposing their position to other adversarial or curious users.

To ensure a comprehensive understanding regarding which data are shared between users, Equation (14) is rearranged as follows:

$$\hat{\boldsymbol{x}}^{(n)} = \underbrace{\boldsymbol{A}^{(n)}\tilde{\boldsymbol{y}}^{(n)}}_{n\text{-th receiver}} - \boldsymbol{A}^{(n)}\underbrace{(\tilde{\boldsymbol{y}}^{(m)} - \boldsymbol{H}^{(m)}\hat{\boldsymbol{x}}^{(m)})}_{m\text{-th receiver}} \tag{18}$$

where:

$$\boldsymbol{y}^{(n,m)} = \tilde{\boldsymbol{y}}^{(n)} - \tilde{\boldsymbol{y}}^{(m)} \tag{19}$$

$$\tilde{\boldsymbol{y}}^{(n)} = \boldsymbol{\rho}^{(n)} + \boldsymbol{H}^{(n)}\mu_0^{(n)} - \mathbf{h}(\mu_0^{(n)}) \tag{20}$$

$$\tilde{\boldsymbol{y}}^{(m)} = \boldsymbol{\rho}^{(m)} + \boldsymbol{H}^{(m)}\mu_0^{(m)} - \mathbf{h}(\mu_0^{(m)}) \tag{21}$$

$$\boldsymbol{A}^{(n)} = (\boldsymbol{H}^{(n)\top}\boldsymbol{R}^{-1}\boldsymbol{H}^{(n)})^{-1}\boldsymbol{H}^{(n)\top}\boldsymbol{R}^{-1} \tag{22}$$

Here, the two 4-dimensional vectors in Equation (18) only carry information of the corresponding user.

Locally, to compute $\boldsymbol{A}^{(n)}$, the n-th user requires knowledge of its $\mathbf{R}^{(n)}$ and the other user's $\mathbf{R}^{(m)}$ covariances, and vice versa. Therefore, this process requires both users to share that information. To alleviate this requirement, an unweighted LS solution, as in Equation (16), could be considered, in which case the matrix can be computed as $\boldsymbol{A}^{(n)} = (\boldsymbol{H}^{(n)\top}\boldsymbol{H}^{(n)})^{-1}\boldsymbol{H}^{(n)\top}$.

Let $\mathbf{b}^{(m)}$ denote the data shared by the m-th user with the n-th user, as shown in Equation (18), with $\mathbf{b}^{(m)} = \tilde{\boldsymbol{y}}^{(m)} - \boldsymbol{H}^{(m)}\hat{\boldsymbol{x}}^{(m)}$. This is the only information that must be transmitted, as shown in Figure 2. As for the dimensions of these parameters, $\mathbf{b}^{(m)}$ is an L-dimensional vector, and $\mathbf{A}^{(n)}$ has dimensions of $4\times L$. The L parameter value is the number of available satellites that are shared between the two users, with $L \geq 4$.

The non-encrypted collaboration shown in Equation (18) does not provide any information regarding the positions of the n-th and m-th user, because no information about the n-th user is ever sent and $\mathbf{b}^{(m)}$ only has residual information from which $\hat{\boldsymbol{x}}^{(m)}$ cannot be inferred, respectively. However, this scheme does leak information regarding the position uncertainty of the m-th user. Potentially, if K instances of that vector $\{\mathbf{b}_k^{(m)}\}_{k=1}^{K}$ were available for a user (the n-th user or an eavesdropping third user), the uncertainty $\mathrm{Cov}(\hat{\boldsymbol{x}}^{(m)})$ could be inferred as $\frac{1}{K}\sum_{k=1}^{K}\mathbf{b}_k^{(m)}\mathbf{b}_k^{(m)\top}$ if $K > 4$. Therefore, while the position information could be preserved, these approaches (i.e., non-encrypted or regular encryption) still leak information about a user to a certain extent through the position uncertainty of the user. This leakage could be undesirable for certain sensitive applications in which this information might need to be kept secret because it could itself leak information about the position of the user (e.g., low uncertainty might indicate that the user is in a good-visibility area, which could help pinpoint its location, or might provide an indication regarding the type of sensors and algorithms on board, which again could potentially be used to determine the user's identity).

4.2 | Encryption Approach

In this scheme, we propose to preserve the access to $\mathbf{b}^{(m)}$ by encrypting its transmission. To consider a secure communication link between the n-th user and m-th user, one approach is to use an encrypted method, where the ability to perform any computation within the encrypted domain is not required (Katz & Lindell, 2007). Using this encrypted method will allow the data shared between users to be encrypted, thus concealing the data transfer between users. The regular encryption can be seen as a message being encrypted by a public key, thus converting a message to become encoded and then again converted to a ciphertext. The decryption aspect would involve a process similar to encryption: with a private key, the ciphertext is decrypted and then decoded to the message that was encrypted (Katz & Lindell, 2007). This process is described in Section 3, with the exception that any computation can be performed on a ciphertext. Using this scheme with the CoPo approach will involve encrypting $\mathbf{b}^{(m)}$:

$$\mathrm{Dec}(\mathrm{Enc}(\mathbf{b}^{(m)})) = \mathbf{b}^{(m)}$$

The procedure for using regular encryption in the CoPo approach will require the n-th user to create the public and private keys, denoted as $\mathbf{Pk}^{(n)}$ and $\mathbf{Sk}^{(n)}$,

respectively. After creating these keys, the n-th user will share its public key with the m-th user. It is important to note that the distribution of the public key $\mathbf{Pk}^{(n)}$ occurs once, and this key is required when any desired data are sent in an encrypted manner to the n-th user. Following the procedure shown in Section 4.1, the n-th user will compute its parameters $\mathbf{A}^{(n)}$ and $\tilde{\boldsymbol{y}}^{(n)}$ locally, while the m-th user will compute its parameter $\mathbf{b}^{(m)}$ locally. Once the m-th user computes its parameters and obtains the public key from the n-th user, $\mathbf{Pk}^{(n)}$, the m-th user encrypts $\mathbf{b}^{(m)}$ with $\mathbf{Pk}^{(n)}$ and sends these encrypted data to the n-th user. It is worth emphasizing that the encryption process entails encrypting each individual element within $\mathbf{b}^{(m)}$, thus generating ciphertexts for each element. A more detailed process is presented in the next subsection. When the n-th user receives the encrypted data, $\mathsf{Enc}(\mathbf{b}^{(m)})$, this user will use its private key, $\mathbf{Sk}^{(n)}$, to decrypt the encrypted data and obtain $\mathbf{b}^{(m)}$, which should be the same as if the m-th user never encrypted it. Similar to the encryption process, the decryption process includes decrypting every ciphertext that represents the element entry of $\mathbf{b}^{(m)}$. With the decrypted data, the n-th user will then proceed with the process of computing its position, as shown in Equation (18), which can be rearranged as follows:

$$\hat{\boldsymbol{x}}^{(n)} = \boldsymbol{A}^{(n)}\tilde{\boldsymbol{y}}^{(n)} - \boldsymbol{A}^{(n)}\mathsf{Dec}(\mathsf{Enc}(\mathbf{b}^{(m)})) \tag{23}$$

As discussed in Section 3, encrypting the data will not have an impact on the data itself; therefore, Equations (18) and (23) are equivalent, while the data transmission is changed, as shown in Figure 2. By implementing an encryption method, any eavesdropping adversarial users will be prevented from inferring the uncertainty on $\hat{\boldsymbol{x}}^{(m)}$ if they intercept $\mathsf{Enc}(\mathbf{b}^{(m)})$. However, this information might still be revealed to the n-th user, as this user has access to $\mathbf{b}^{(m)}$ as in the non-encrypted scheme.

4.3 | Homomorphic Encryption Approach

The third alternative scheme implementing Equation (18) aims at preserving the privacy of $\mathbf{b}^{(m)}$ from not only an eavesdropper but also the cooperative user n. An alternative approach for ensuring the privacy of sensitive information among cooperative users involves the utilization of homomorphic encryption. Using homomorphic encryption will allow addition and multiplication operations to be performed on ciphertexts, the encrypted data, without leaking any sensitive information. Similar to the previous two approaches, the implementation of homomorphic encryption will yield identical results, as the exact same formula (Equation (18)) is applied. The only difference is in the shared information in which the user computes certain elements, as shown in Figure 2.

The key distribution is similar to that in the encryption approach shown in Section 4.2. The n-th user creates a set of public and private keys, denoted as $\mathbf{Pk}^{(n)}$ and $\mathbf{Sk}^{(n)}$, respectively. The n-th user will proceed in sharing its public key with the m-th user. As an overview, the m-th user will be tasked with solving Equation (14) in an encrypted manner without leaking information about the n-th user. Once the task is completed by the m-th user, the m-th user returns the encrypted results to the n-th user, who is then able to decrypt and compute $\hat{\boldsymbol{x}}^{(n)}$. As a consequence, when implementing a homomorphic encryption scheme with the following:

$$\mathsf{Enc}(\boldsymbol{A}^{(n)} \cdot \mathbf{b}^{(m)})_{\mathbf{Pk}^{(n)}} = \mathsf{Enc}(\boldsymbol{A}^{(n)})_{\mathbf{Pk}^{(n)}} \cdot \mathsf{Enc}(\mathbf{b}^{(m)})_{\mathbf{Pk}^{(n)}} \tag{24}$$

where $\mathbf{b}^{(m)}$ is the same parameter used in the previous two methods, this process will actually solve for the user's dependent calculations in Equation (14).

After the n-th user transfers its public key $\mathbf{Pk}^{(n)}$ to the m-th user, the m-th user will encrypt its data, $\mathbf{b}^{(m)}$, using the public key $\mathbf{Pk}^{(n)}$. Additionally, the n-th user also encrypts its data, $\mathbf{A}^{(n)}$, and sends these encrypted data to the m-th user, as shown in Figure 2. Before the encryption process, each element entry of $\mathbf{A}^{(n)}$ and $\mathbf{b}^{(m)}$ is first encoded to a fixed-point representation. As the number of bits allocated to the integer and fraction part increases, the value becomes more precise. In Figure 3, the encoder result is represented by y, where the positive exponent value represents the allocated bits for the integer and the negative exponent values represent the allocated bits for the fraction portion. After the encoding process, each value is then converted to a plaintext polynomial, x^n+1. Once in the plaintext polynomial form, the polynomial is converted to a ciphertext by the encryption process. The decryption will follow the inverted process.

The m-th user will perform the computation in Equation (24) locally. As previously stated in Section 4.1, $\mathbf{b}^{(m)}$ is an L-dimensional vector and $\mathbf{A}^{(n)}$ is a $4\times L$ matrix when they are not encrypted, and L is the number of available satellites shared between the two users. When the two users share the same four observed satellites (i.e., $L=4$), there is a risk that the n-th user will be able to obtain the position uncertainty of the m-th user by simply sending $\mathbf{A}^{(n)}=\mathbf{I}$ and receiving $\mathbf{b}^{(m)}$ directly in return. Therefore, to avoid this risk, the number of shared observed satellites must be greater such that $L>4$. Once the number of shared observed satellites is greater than 4, then the n-th user would not be able to obtain $\mathbf{b}^{(m)}$ from $\mathbf{A}^{(n)}\mathbf{b}^{(m)}$ because this is an undetermined system in which L unknowns are to be inferred from 4 measurements. Therefore, because $\mathbf{A}^{(n)}$ is not a full-column rank matrix $\text{rank}(\mathbf{A}^{(n)})\leq 4<L$, it yields to infinitely many solutions if $(\mathbf{A}^{(n)})^{-1}(\mathbf{A}^{(n)}\mathbf{b}^{(m)})$ is attempted. This fact protects the m-th user against the n-th user knowing its position uncertainty. Thus, homomorphic encryption becomes a feasible option, and the operations that the m-th user performs within the encryption domain are supported by the BFV scheme, as it can perform the required multiplication and addition operations.

After computing Equation (24), the m-th user then returns the encrypted results to the n-th user, as shown in Figure 2. When the n-th user receives the encrypted results, using its secret key $\mathbf{Sk}^{(n)}$, the n-th user decrypts the results and solves for

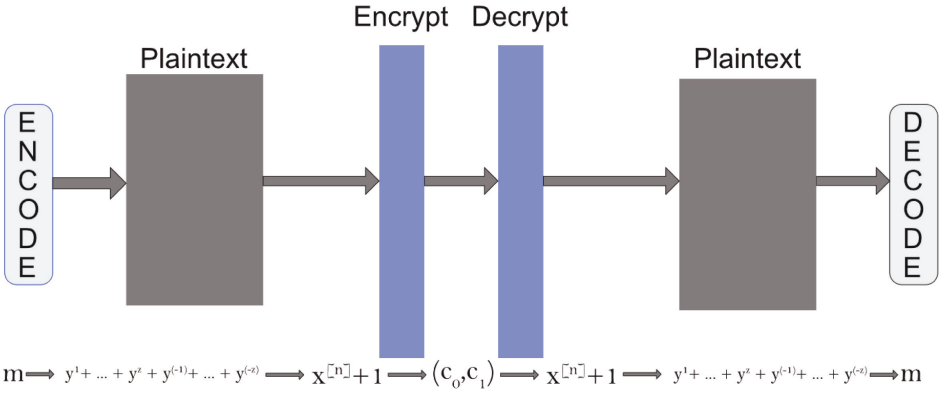

FIGURE 3 For a message m to be encrypted, it must first be encoded. Once the value is encoded, the encoded value is converted into a plaintext polynomial within the plaintext polynomial ring R_t. Afterwards, this plaintext can be converted into a ciphertext. To decrypt the ciphertext, a similar process is applied: the ciphertext is first decrypted into a plaintext and then decoded to recover the message.

Equation (14). In Figure 3, the ciphertext results are reverted back to a plaintext polynomial form (x^n+1) via the decryption algorithm and the appropriate secret key. This plaintext polynomial is then sent to the decoder, which decodes y. The decoded outcome is the original plaintext form. After the process is complete, the results should be the same as if all of the computations were performed via an non-encrypted method. A similar approach is required to calculate Equation (12).

Lastly, the complexities of the two encryption schemes are considered and compared with that of the non-encrypted scheme. The non-encrypted scheme must perform the same number of operations, i.e., $\sum_{\ell} \boldsymbol{A}^{(n)}[j,\ell]\cdot \mathbf{b}^{(m)}[\ell]$. This will lead to a complexity of $\mathcal{O}(L)$. Here, the worse-case scenario was used because the computation will require the use of all of the observed satellites, i.e., L. The complexity for the regular encryption scheme will be the same because, after $\mathsf{Enc}(\mathbf{b}^{(m)})$ is decrypted, the computation requisite will be identical, as in the first scheme. As for the homomorphic encryption approach, the operation first takes into consideration the polynomial degree value, which is the same for both the plaintext and ciphertext. Secondly, the number theoretic transform (NTT) is used to perform the polynomial multiplication operation. Using this method will allow the multiplication of a polynomial to have a complexity of $\mathcal{O}(nlogn)$ (Harvey, 2014). Thirdly, the number of polynomials considered in a ciphertext, i.e., $\log q$, also impacts the complexity. With the NTT method and considering the number of observed satellites between the users, each user's computation complexity is $\mathcal{O}(Ln\log n(\log q)^2)$.

In terms of transmitting data between users, the non-encrypted approach will require the least amount of data compared with the other approaches. In this approach, only the $\boldsymbol{b}^{(n)}$ parameter is transmitted; therefore, the storage complexity will be $\mathcal{O}(L)$, strictly depending on L. For the regular encryption approach, the encrypted $\boldsymbol{b}^{(n)}$ parameter is transmitted. Because the element entry within $\boldsymbol{b}^{(n)}$ has a ciphertext size of polynomial degree n and a $\log q$ number for the ciphertext, the message will have a storage complexity of $\mathcal{O}(Ln\log q)$. The result for the homomorphic encryption approach is similar to that of the encryption approach, with the exemption of the $\boldsymbol{A}^{(n)}$ parameter. When the n-th user transmits its encrypted $\boldsymbol{A}^{(n)}$, it originally has $4\times L$ elements, with each having its own ciphertext. Therefore, a message would have a storage complexity of $\mathcal{O}(Ln\log q)$. The worse-case scenario is evaluated for all of the approaches, as they will require all of the element entries of $\boldsymbol{b}^{(n)}$ for the first approach and the encrypted elements for the second and third approaches to be transmitted.

4.4 | Attack Models

The three models address the privacy of collaborating users against any adversarial user not participating in the collaborative framework and also among participating collaborative users. From the prospective of when an adversarial user desires to obtain information from any of the participating users, the privacy approach will distinguish between the non-encryption method and the two encryption methods. For the non-encryption method, the adversarial user is capable of obtaining access to the data being transmitted between the two users. The issue for the adversarial user is that the information exchange between the n-th user and m-th user will be of no value if its objective is to obtain any valuable positioning information for either user. Because $\mathbf{b}^{(m)}$ is the only information that is made public, the adversarial user will only obtain the information of the m-th user's position uncertainty. Therefore, the confidentiality of the m-th user's position is maintained, eliminating the potential for any adversarial user or the n-th user to access that information.

Furthermore, the n-th user's position information will not experience any risk from this adversarial user, as no information from the n-th user is shared or made public.

In the regular encryption and homomorphic encryption schemes, the privacy is enhanced such that no information is leaked to any adversarial user outside of the framework. Suppose the m-th user sends the n-th user its encrypted information, $\mathsf{Enc}(\boldsymbol{b}^{(m)})_{\mathbf{Pk}^{(n)}}$, and an adversarial user intercepts this encrypted message. The adversarial user will receive an encrypted message and will not be able to decrypt the message without the n-th user's private key. Similarly, when the homomorphic encryption approach is employed, an adversarial user could potentially access the encrypted solution transmitted by the m-th user to the n-th user, $\mathsf{Enc}(\boldsymbol{A}^{(n)})_{\mathbf{Pk}^{(n)}} \cdot \mathsf{Enc}(\mathbf{b}^{(m)})_{\mathbf{Pk}^{(n)}}$, and the adversarial user will not be able to decrypt the message for the same reason. As a result, the n-th user and m-th user data will remain private at all times against any adversarial user that attempts to infer sensitive information (e.g., their locations) based on communications between the two users. There is also the consideration of an adversarial user that poses as a collaborating user, which follows the latter privacy concern discussed below.

Let us assume the case in which the m-th user is an adversarial user and establishes a connection with the n-th user. The adversarial user seeks to obtain more information about the n-th user's location or other valuable information that may help in determining the location of the n-th user. The argument is the same as that for an adversarial user who does not participate in the collaborative approach when the non-encryption method is used. Because the n-th user does not make any of its information public, it will never risk having its information compromised. For the regular encryption scheme, the argument is the same: the n-th user never shares any information with the m-th user. For the homomorphic encryption scheme, the n-th user transmits $\mathbf{A}^{(n)}$ in an encrypted manner; however, because the n-th user is the sole holder of the private key that is needed to decrypt any encrypted information and the private key is never distributed, the possibility that the m-th user will decrypt the ciphertext and obtain the message within is negligible.

We also consider the case in which the n-th user is an adversarial user. The non-encryption scheme will have the same argument as before when the transmitted information is already made public; therefore, when an adversarial user disguises itself as the n-th user, the situation is equivalent to the case of an external user that attempts to infiltrate the collaborative framework. For the two encryption methods, the n-th user will have access to the private key and to all of the encrypted data. In the regular encryption approach, the adversarial user will proceed in creating a set of keys and will share them with the m-th user. The m-th user will encrypt its data and send the data to the adversarial user, who poses as the n-th user. With the private key, the n-th user will obtain the position uncertainty of the m-th user. As stated before, this will not provide the adversarial user any valuable information regarding the position of the m-th user.

For the homomorphic encryption case, the n-th user (the adversarial user) follows the original procedure, as described above: the n-th user sends the m-th user its encrypted data, and the the m-th user performs the encrypted calculations in Equation (24) and returns the encrypted results to the n-th user. Because this approach follows the constraint of $L > 4$, after the n-th user decrypts the results, the n-th user will not be able to determine the correct values of $\boldsymbol{b}^{(m)}$. Therefore, the n-th user (the adversarial user) will not obtain any valuable information about the m-th user's location. Still, the m-th user will need to encrypt its information, $\boldsymbol{b}^{(m)}$, using the n-th user's public key because the m-th user is required to perform the encrypted operations in Equation (24). Although the m-th user always assumes that the n-th user is not an adversarial user, for both encryption methods, the

TABLE 1
Summary Table of the Information That Can Be Potentially Leaked From User m to Either an Eavesdropping Agent or the n-th User

	Eavesdropper privacy		n-th user privacy		
Scheme	$\hat{\boldsymbol{x}}^{(m)}$	$\text{Cov}(\hat{\boldsymbol{x}}^{(m)})$	$\hat{\boldsymbol{x}}^{(m)}$	$\text{Cov}(\hat{\boldsymbol{x}}^{(m)})$	**RT**
non-encryption	✓	✗	✓	✗	✓
regular encryption	✓	✓	✓	✗	✓
homomorphic encryption	✓	✓	✓	✓	✗

Note: The primary information indicates either the user's position $\hat{\boldsymbol{x}}^{(m)}$ or the associated position uncertainty $\text{Cov}(\hat{\boldsymbol{x}}^{(m)})$. A check denotes that the quantity is kept private, while a cross refers otherwise. The table also shows the real-time (RT) readiness of each approach, given the corresponding complexity.

m-th user can be confident that valuable information about its location will not be exposed in the case that the n-th user is an adversarial user.

A summary of the privacy-preserving capabilities of the three schemes is provided in Table 1, which shows an increasing level of privacy and implementation complexity from non-encryption as the simplest scheme to homomorphic encryption as the scheme that arguably requires the most resources.

5 | RESULTS

This section presents an analysis of the non-encrypted and encryption-based CoPo schemes, as well as a comparison with the classical non-cooperative PVT solution implemented as iterative LS and WLS algorithms. The objectives of this analysis were *i*) to verify that the cooperative (DGNSS) scheme indeed reduces the impact of ionospheric delays on pseudoranges even when the encryption scheme is employed, as opposed to the benchmark non-cooperative solution, *ii*) to gain an understanding of the relevant parameters in the homomorphic encryption scheme, as utilized in the CoPo solution, *iii*) to demonstrate the use of the homomorphic encryption scheme and validate its performance and privacy features, and *iv*) to demonstrate the use of these methods with real-world data. The subsections below address these objectives.

5.1 | CoPo Scheme

A simulator that replicates an open-sky environment in which users are able to obtain GNSS observable measurement values without any obstruction or multipath was used. The benchmark was simulated to obtain noisy observable pseudorange measurements, with a standard deviation of approximately 10 m, and observation values from 30 available satellites were utilized. The simulator was set to observe this large amount of satellites (rarely seen in practical DGNSS setups) in order to account for the case in which both users observe the maximum number of possible satellites, thus leading us to explore the maximum number of values that must be set as private and of the computations needed in the encrypted domain. This case was applied to further understand the impact of the operations performed in the encrypted domain for the homomorphic encryption scheme. Furthermore, an evaluation was conducted to assess the impact of the ionospheric effect on the measurement values. For the (ionosphere-free) WLS algorithm case, the ionospheric delay was removed.

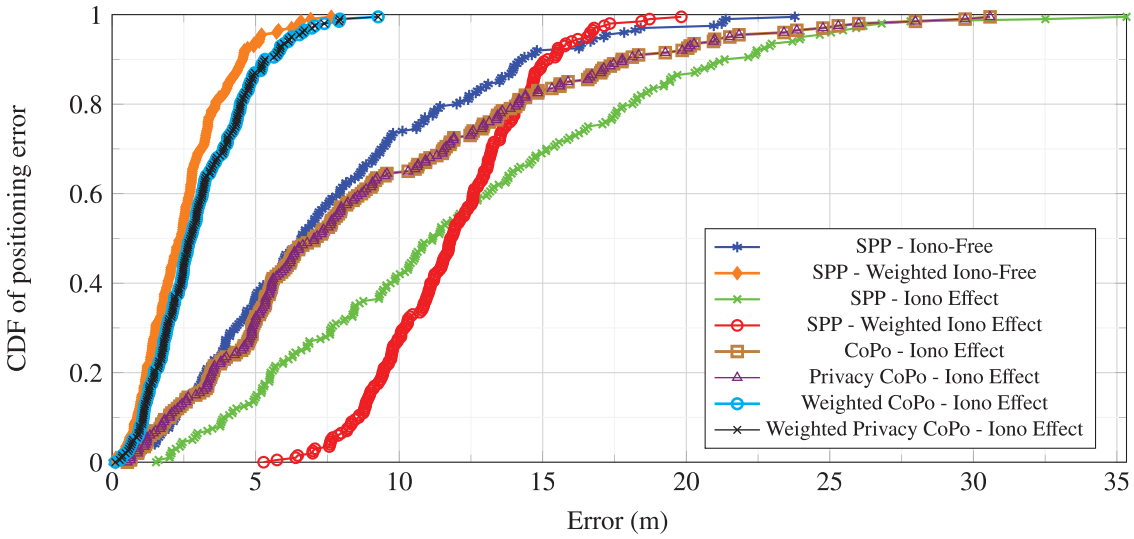

FIGURE 4 Positioning error comparison of the benchmark non-cooperative scheme with and without ionospheric disturbances and the proposed cooperative scheme with and without privacy layers

Figure 4 shows the cumulative density function (CDF) of the positioning error for all of the methods mentioned above: *i*) The benchmark method, that is, the case with the WLS (ionosphere-free) non-CoPo solution. This corresponds to the weighted standard point positioning (SPP) method. *ii*) The same SPP non-cooperative WLS method with ionospheric perturbations. Based on Figure 4, the ionospheric delay skews roughly 90% of the PVT quantities by approximately an additional 10 m. The objective was to reduce the ionospheric effect by using a cooperative scheme and produce PVT values that mirrored the PVT values seen within the benchmark case. *iii*) The SPP non-cooperative LS method with and without the ionospheric error effect. These solutions assumed that the noise variance was the same for the observed values, thus reducing the accuracy of the results. *iv*) The case of the CoPo scheme (without a privacy-preserving scheme) with WLS and LS algorithms in the presence of ionospheric delays. The scheme with the WLS algorithm provided values similar to those in the benchmark scenario, virtually eliminating the ionospheric error effects. *v*) The implementation of the privacy-preserving schemes under ionospheric errors. Given that the non-encryption and regular encryption approaches did not have a direct impact on the computation of the CoPo aspect, the results were the same as the CoPo results (with no privacy-preserving scheme) and, to some extent, to the results obtained when the CoPo scheme incorporated homomorphic encryption. The privacy-preserving scheme that included homomorphic encryption had the same results, depending on its encryption parameters. Here, the encryption parameters were set as follows: a polynomial degree n of 4096, a security parameter λ of 128, and a plaintext modulus value t of 5003. Using these encryption parameters for the homomorphic encryption allowed the ciphertext values to stay within the noise threshold (thus being decryptable) when L was set to 30. This indicates that 30 satellites were equally observed by the users.

As analyzed earlier in Section 2, the ionospheric error delay can be reduced when the pseudoranges measured by two receivers (n-th and m-th users) are subtracted from one another in the CoPo scheme. As briefly mentioned, in contrast to the WLS approach, the LS approach assumes that the measurement noise variance is the same, which caused the results to be less accurate. This result was seen in both the cooperative and non-cooperative approaches. This finding indicates that

if users are able to obtain this information, then the accuracy will be improved, but this may not always be the case.

In the analysis of the three privacy-preserving schemes within the CoPo scheme, the results were expected to replicate those of the standard CoPo scheme. Therefore, we labeled these results as "Privacy CoPo" in the plots, as they are equivalent with regard to positioning performance. With the non-encryption privacy-preserving scheme, the results were exactly the same, as nothing was truly altered. For the regular encryption scheme, the approach simply hid the message within a ciphertext; the message consisted of the $\mathbf{b}^{(m)}$ quantities sent by the m-th user to the n-th user. After receiving the encrypted message, the n-th user decrypted the message, but the process of determining the estimated PVT quantites of both users remained the same as the standard CoPo solution. The homomorphic encryption scheme instituted a caveat whenever the ciphertext exceeded the permissible noise threshold. Whenever this situation occurs, the outcome of the decryption process of that ciphertext will decrypt a value differing from the original value. Therefore, this approach requires a further analysis to be explored, with the impact that the encrypted parameters enable the homomorphic encryption to produce reliable estimated PVT values.

5.2 | Homomorphic Encryption Parameters

Although the homomorphic encryption CoPo method obtained results similar to the benchmark and the CoPo method, this did not hold true for lower polynomial degree values (n), with higher security parameters (λ) and high plaintext modulus values (t). As mentioned above and as shown in Figure 4, the privacy-preserving CoPo method will produce the same results as the benchmark, as long as the ciphertext does not exceed the noise threshold. The ciphertext noise level was analyzed via the Pyfhel library for different encrypted parameters (Ibarrondo & Viand, 2021). Two polynomial degree values were used: $n = 4096$ and $n = 8192$. Values higher than 8196 added more than enough computational capacity and increased the storage complexity, as discussed in Section 3, to perform all of the computations needed for the CoPo scheme with the implementation of homomorphic encryption to satisfy the noise constraints of the encryption method.

In Figure 5, the noise level results are shown to reach the noise threshold for different security parameters (λ), while the polynomial degree $(n = 4096)$ remains the

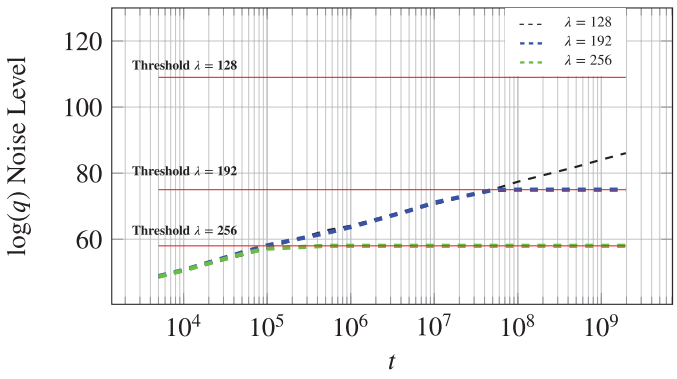

FIGURE 5 Increase in noise level for different values of t and λ for a polynomial degree value $n = 4096$

When the noise level reaches the threshold for a specific configuration, the ciphertext cannot be correctly deciphered. For large security parameter values λ, there is a higher likelihood of this case arising as the modulus value increases.

same for all cases. As observed earlier, as the security parameter increases (indicating an increase in the security of the ciphertext) and the plaintext modulus value increases, the capacity to perform a certain number of operations in the encryption domain begins to decrease. As discussed in Section 3, increasing the plaintext space or allowing for higher coefficient values will expand the plaintext ring, therefore increasing the number of polynomials to which an encoder can be mapped. This feature becomes important during the encoding process. The downside of having high plaintext coefficient values is the increased noise level within the ciphertext. The ciphertext modulus value depends on the security parameter; increasing λ will limit the amount of operations that can be performed on the encrypted message. Based on Figure 5, the ciphertext's noise level increases sublinearly. It can be observed that the noise threshold is the lowest for a security parameter value of 256 compared with the other two values. This result indicates that the coefficient modulus value of the ciphertext is much smaller. Decreasing this value limits the ciphertext coefficient values; if operations are repeatedly performed on these coefficient values, the coefficients will reach their upper bound. It is ideal to operate under the ciphertext's noise threshold in order to maintain a correct decryption. For a security parameter value of 128, the ciphertext's noise level did not reach the ciphertext's noise threshold, indicating that all of the polynomial values in this domain are valid for correctly decrypting the message within the ciphertext.

Similar to Figure 5, the ciphertext's noise level was also analyzed and compared with its noise threshold as the polynomial degree increased to 8192 (refer to Figure 6). It can be observed that as the polynomial degree increased, the noise threshold of the ciphertext also increased. It is important to note that the noise increase rate remained the same compared with the case in which the polynomial degree value was 4096, giving a reassurance that a higher polynomial degree can be a solution toward correctly decrypting the message within a ciphertext. The results shown in Figure 6 capture this evidence, as the ciphertext's noise level remained below the noise threshold, with the same plaintext modulus domain. The ciphertext noise level was also analyzed for a polynomial degree value of 16384. The noise level was consistent with the other two cases; the differences in the ciphertext noise thresholds are summarized in Table 2.

Although increasing the polynomial degree increases the privacy of the ciphertext, this comes with a computational complexity cost. Table 3 provides a summary of the time consumption for different polynomial degree values. In particular, the table shows the average execution time of the homomorphic encryption applied

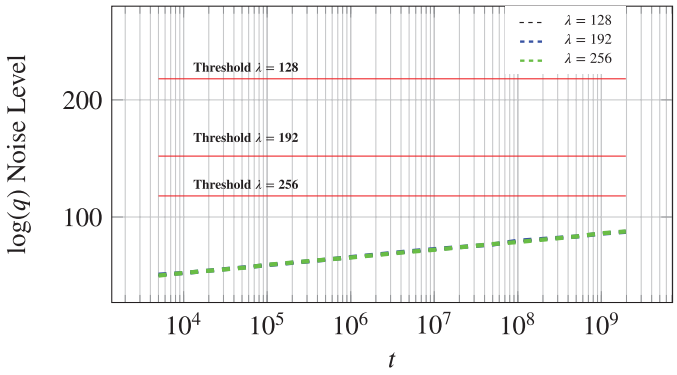

FIGURE 6 Increase in noise level for different values of t and λ for a polynomial degree value of $n = 8192$
Compared with $n = 4096$, the noise threshold increases, which enables more encrypted operations to be computed.

TABLE 2
Noise Threshold Results For Different Security Parameters and Polynomial Degree Values

Security Param (λ)	Polyn Degree (n)	Noise Threshold log(q)
128	4096	109
	8192	218
	16384	438
192	4096	75
	8192	152
	16384	300
256	4096	58
	8192	118
	16384	237

TABLE 3
Average Execution Time of Homomorphic Encryption Applied to the LS Algorithm When $L = 6$ and $L = 30$ Satellites Are Observed by Each User

Number of Satellites (L)	Polyn Degree (n)	Average Execution Time (s)
6	2048	0.0491153
	4096	0.1296411
	8192	0.4144107
	16384	1.6944951
30	2048	2.48926928
	4096	7.53169804
	8192	25.07900775
	16384	93.36147592

Note: The test was performed for four polynomial degree n values: 2048, 4096, 8192, and 16384. The security parameter $\lambda = 128$ and plaintext modulus value $t = 5003$ remained constant for all cases.

to the LS algorithm in which the two users obtained their own pseudorange measurements for $L = 30$ satellites (clearly an upper bound on the number of typically considered satellites) and $L = 6$ satellites. Recall that the LS algorithm is an iterative process (approximately 5 iterations are typically sufficient for convergence); thus, the results show the average execution time until convergence. The algorithm was implemented in Python and run on an Intel(R) Core(TM) i7-8650U CPU @ 1.90GHz platform; therefore, it is expected that a more optimized implementation would improve the execution time metrics.

The polynomial degree will not only expand the plaintext polynomial, but will also increase the ciphertext polynomial, which, in turn, increases the ciphertext size, thus increasing the computational complexity. The trade-off is as follows: on one hand, increasing the polynomial degree with a high security parameter will increase the computational complexity, which will increase the number of operations in the encrypted domain. In terms of security, this will demand that the ciphertext modulus be smaller. On the other hand, maintaining a small polynomial degree with a high security parameter will increase the likelihood of having a corrupted message. Alternatively, having a small polynomial degree with a lower security parameter will decrease the number of operations in the encrypted domain while still decrypting the message correctly, as evidenced by Figure 5.

5.3 | Homomorphic Encryption CoPo Scheme

In addition to analyzing the ciphertext noise level and comparing it with the noise threshold, the effects of the noise level on the CoPo scheme with the implementation of the homomorphic encryption scheme were also analyzed. Using a range of prime values for the plaintext modulus parameter and the additional encryption described above, the privacy-preserving CoPo scheme with homomorphic encryption provided the same results as the benchmark case. Table 4 provides a summary of these results.

When the noise level was below the noise threshold, indicating a correct decryption, the CDF values produced the same results as shown in Figure 4. From Table 4, this success is represented within the plaintext modulus values. For example, when the polynomial degree value was set to 4096 and the security parameter was set to 192, valid plaintext modulus values were below 100003. Values below this bound produced the same results from the benchmark case, up to $t = 5003$. A value below 5003 caused rounding of the input message, resulting in reduced accuracy of the WLS and LS estimate values, whereas the same number of iterations per epoch as the benchmark case was needed to converge to the estimated PVT quantity. In contrast, when the noise level was above the noise threshold, the message within the ciphertext produced random values, causing the estimated PVT quantities to become random. Because each PVT quantity depended on the previous estimated quantity, the cascade of error continued to produce incorrect estimated values.

5.4 | Experiments With Real-World Data

Upon obtaining the simulated results, we evaluated the performance of the privacy-preserving CoPo scheme, specifically the homomorphic encryption scheme, using data collected from two Android phones. These two devices acted as the two users in the CoPo scheme. In the experimental setup, the two devices were separated by approximately 321.9 m in an outdoor environment at the Northeastern University campus, Boston MA, as shown in Figure 7. Table 5 provides the recorded C/N_0 values for each user across their common satellite IDs.

The GPS Measurement Tools were used to read and process data from the GNSS Logger App (Banville & Van Diggelen, 2016). The raw data were processed to obtain the following necessary information: the satellite ID list, time of reception, time of transmission, pseudoranges for each full time cycle of measurements, pseudorange error estimates, pseudorange rate, and pseudorange rate error estimates. With the possibility that each user observed different satellites and obtained invalid values

TABLE 4
Valid Plaintext Modulus Ranges for the Values of n and λ Considered Here

Polyn Degree (n)	Security Param (λ)	Plaintext Mod (t)
4096	128	$5003 \leq t \leq 1000000207$
	192	$5003 \leq t \leq 10000439$
	256	$5003 \leq t \leq 100003$
8192	128	$5003 \leq t \leq 1000000207$
	192	$5003 \leq t \leq 1000000207$
	256	$5003 \leq t \leq 1000000207$

FIGURE 7 Geographic satellite image depicting the true locations of the n-th and m-th users for the setup used in the real-world data experiment at the Northeastern University campus, Boston MA
The two users were separated by approximately 321.9 m.

TABLE 5
Minimum and Maximum C/N_0 Values Observed by the Two Collaborative Users in the Real-World Data Experiment

Satellite ID	n-th User [Min - Max] (dB-Hz)	m-th User [Min - Max] (dB-Hz)
3	[26.8 – 42.2]	[23.5 – 46.5]
16	[16.8 – 28.7]	[16.9 – 27.5]
25	[23.8 – 28.7]	[16.8 – 23.5]
26	[21.4 – 38.2]	[18.2 – 32.1]
31	[21.8 – 43.1]	[29.1 – 44.4]
32	[20.4 – 46.7]	[35.6 – 46.7]

Note: The users had six common satellites used in the CoPo scheme.

throughout the full cycle time of measurement, the processed data were synchronized to have the same observed satellites. To synchronize the data between receivers, the shared satellite IDs were identified for each full cycle time of measurement. Interpolation was used to account for any missing measurements, and invalid values were discarded. Furthermore, to obtain satellite information such as the transmission time, ephemeris files from the National Aeronautics and Space Administration's archive of space geodesy data were retrieved. As a comparison method, the ionospheric delay was computed based on the ionospheric coefficients broadcast within the navigation message and the Klobuchar model, which was then used to remove that error from the pseudorange measurements. These results are labeled as "Iono-Free" in Figure 8.

Following a process similar to that for the simulator, the locations of both users were determined by using the privacy-preserving CoPo scheme and compared with the results of other positioning approaches. Figure 8 summarizes the results of the estimated position values for each of the methods: non-cooperative LS and WLS; ionosphere-free non-cooperative LS and WLS, where Klobuchar modeling was used; cooperative LS and WLS; and the privacy-preserving method in which

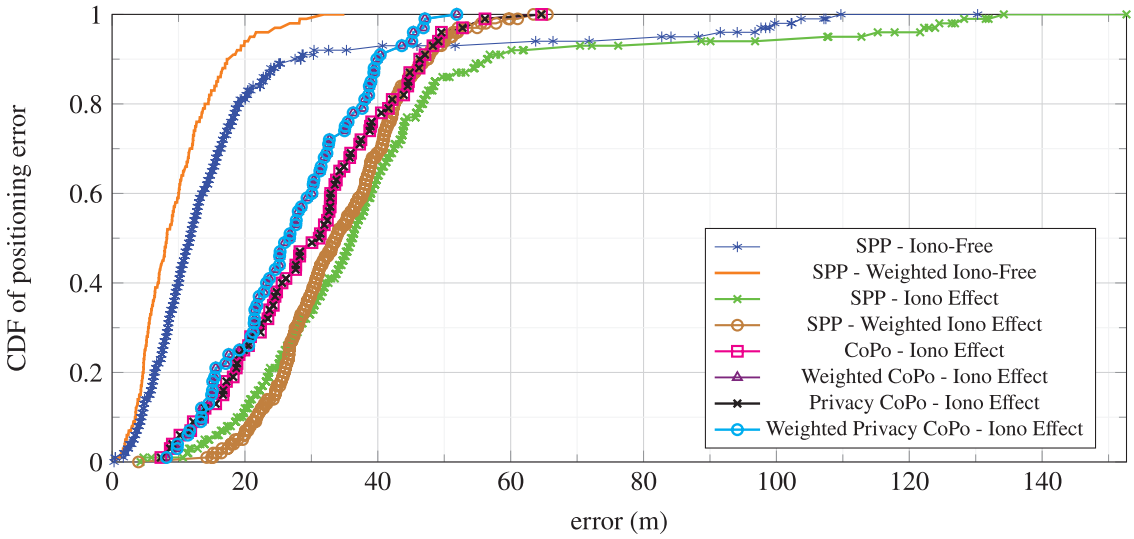

FIGURE 8 CDF of the positioning error obtained by using raw data collected from Android smartphones
The case study compared non-cooperative LS and WLS (ionosphere-free), non-cooperative LS and WLS, cooperative LS and WLS, and the implementation of homomorphic encryption within the cooperative WLS and LS methods.

homomorphic encryption was implemented within the cooperative LS and WLS methods. Only the LS method was used in the homomorphic encryption domain because sharing the measurement variance of the users may raise privacy concerns. The results in Section 5.1 indicated that the privacy-preserving cooperative scheme, both with the non-encryption and regular encryption approaches, produced the same outcomes as the homomorphic encryption scheme when the ciphertext noise level was beneath its noise threshold. Therefore, homomorphic encryption was considered here. For the homomorphic encryption scheme, the encryption parameters were set to $t = 5003, \lambda = 128,$ and $n = 4096$. Similar to the simulated-data results, the privacy-preserving CoPo scheme with homomorphic encryption produced the same results as the cooperative algorithm (with no privacy-preserving scheme implemented). Additionally, the time complexity was similar to the case of $n = 2048,$ as shown in Table 3. With computing performance constantly improving, this proof of concept may become practical in the near future. Nevertheless, the validity of this framework with current computing devices is limited to non-real-time and post-processing operations.

6 | CONCLUSIONS

In this paper, multiple approaches were described for addressing the privacy concern that arises when GNSS observables are shared between two receivers. When these two observables are combined together in the standard differential method, ionospheric errors are removed, which allows a more accurate knowledge for the receivers (n-th and m-th users) to compute their position. This scheme falls within the set of DGNSS or cooperative GNSS methodologies. The main concern arising from such collaborative schemes is the privacy leakage by which one could learn where other users are located through shared data. In this paper, three different privacy-preserving schemes are analyzed: a non-encryption scheme, a regular encryption scheme, and a homomorphic encryption scheme. The first scheme

maintains the user positions as private at all times, but exposes the user position uncertainty to eavesdropping adversarial users. The regular encryption scheme limits this exposure to only collaborating users. Still, the implementation of homomorphic encryption removes these two risks, but comes with a large computational consumption.

When these computational operations are performed within the encryption domain created by the homomorphic encryption scheme, the ciphertext noise must be taken into consideration to successfully decipher the message. Such noise is inherently added by the homomorphic encryption scheme and increases as operations are performed on the ciphertext. If the ciphertext noise level reaches a predetermined noise threshold level, the message will be corrupt, which will prevent decryption of the message. To reduce the likelihood of reaching the noise threshold, a set of parameters (i.e., the polynomial degree, security parameter, and plaintext modulus) must be considered in the system design. As shown by simulated and real-world data results, correct decryption comes with a high computational complexity cost. Ideally, smaller parameter values result in less computational complexity. Table 4 provides values that allow for correct decryption.

Indeed, homomorphic encryption may currently be computationally demanding and impractical for real-time applications. This approach could be considered valid for non-real-time applications, such as those involving observables in post-processing or tracking non-critical assets. The research outlook shows that this approach could be readily available for more latency-adverse applications when advances in computing devices enable FHE and related solutions to be obtained more rapidly (Dampf et al., 2015). Alternatively, the other schemes discussed in this paper (non-encrypted and standard encryption) could be considered for latency-critical applications, although, as discussed, their privacy features are reduced compared with the homomorphic scheme.

ACKNOWLEDGMENTS

This work has been partially supported by the National Science Foundation under Awards ECCS-1845833 and CCF-2326559.

REFERENCES

Acar, A., Aksu, H., Uluagac, A. S., & Conti, M. (2018). A survey on homomorphic encryption schemes: Theory and implementation. *ACM Computing Surveys (CSUR)*, *51*(4), 1–35. https://doi.org/10.1145/3214303

Alam, N., & Dempster, A. G. (2013). Cooperative positioning for vehicular networks: Facts and future. *IEEE Transactions on Intelligent Transportation Systems*, *14*(4), 1708–1717. https://doi.org/10.1109/TITS.2013.2266339

Alsamhi, S. H., Ma, O., Ansari, M. S., & Almalki, F. A. (2019). Survey on collaborative smart drones and internet of things for improving smartness of smart cities. *IEEE Access*, *7*, 128125–128152. https://doi.org/10.1109/ACCESS.2019.2934998

Amin, M. G., Closas, P., Broumandan, A., & Volakis, J. L. (2016). Vulnerabilities, threats, and authentication in satellite-based navigation systems [scanning the issue]. *Proc. of the IEEE*, *104*(6), 1169–1173. https://doi.org/10.1109/JPROC.2016.2550638

Banville, S., & Van Diggelen, F. (2016). Precise positioning using raw GPS measurements from android smartphones. *GPS World*, *27*(11), 43–48. https://www.gpsworld.com/innovation-precise-positioning-using-raw-gps-measurements-from-android-smartphones/

Borre, K., Akos, D. M., Bertelsen, N., Rinder, P., & Jensen, S. H. (2007). *A software-defined GPS and Galileo receiver: A single-frequency approach*. Springer Science & Business Media. https://doi.org/10.1007/978-0-8176-4540-3

Brakerski, Z. (2012). Fully homomorphic encryption without modulus switching from classical gapsvp. *Proc. of the 32nd Annual Cryptology Conference (CRYPTO 2012),* Santa Barbara, CA, 868–886. https://doi.org/10.1007/978-3-642-32009-5_50

Calatrava, H., Medina, D., & Closas, P. (2023). Massive differencing of GNSS pseudorange measurements. *Proc. of the 2023 IEEE/ION Position, Location and Navigation Symposium (PLANS),* Monterey, CA, 891–896. https://doi.org/10.1109/PLANS53410.2023.10139935

Chen, L., Thombre, S., Järvinen, K., Lohan, E. S., Alén-Savikko, A., Leppäkoski, H., Bhuiyan, M. Z. H., Bu-Pasha, S., Ferrara, G. N., Honkala, S., Lindqvist, J., Ruotsalainen, L., Korpisaari, P., & Kuusniemi, H. (2017). Robustness, security and privacy in location-based services for future IoT: A survey. *IEEE Access*, *5*, 8956–8977. https://doi.org/10.1109/ACCESS.2017.2695525

Closas, P., Luise, M., Avila-Rodriguez, J. A., Hegarty, C., & Lee, J. (2017). Advances in signal processing for GNSSs [From the Guest Editors]. *IEEE Signal Processing Magazine*, *34*(5), 12–15. https://doi.org/10.1109/MSP.2017.2716318

Dampf, J., Pany, T., Bär, W., Winkel, J., Stöber, C., Fürlinger, K., Closas, P., & Garcia-Molina, J. (2015). More than we ever dreamed possible: Processor technology for GNSS software receivers in the year 2015. *Inside GNSS*, *10*(4), 62–72. https://insidegnss.com/more-than-we-ever-dreamed-possible/

Dardari, D., Closas, P., & Djurić, P. M. (2015). Indoor tracking: Theory, methods, and technologies. *IEEE Transactions on Vehicular Technology*, *64*(4), 1263–1278. https://doi.org/10.1109/TVT.2015.2403868

Dardari, D., Falletti, E., & Luise, M. (2011). *Satellite and terrestrial radio positioning techniques: A signal processing perspective*. Academic Press. https://doi.org/10.1016/C2009-0-61856-0

Fan, J., & Vercauteren, F. (2012). Somewhat practical fully homomorphic encryption. *Cryptology ePrint Archive*. https://ia.cr/2012/144

Feng, T., Wong, W. C., Sun, S., Zhao, Y., & Zhang, Z. (2019). Location privacy preservation and location-based service quality tradeoff framework based on differential privacy. *Proc. of the 16th Workshop on Positioning, Navigation and Communications (WPNC),* Bremen, Germany, 1–6. https://doi.org/10.1109/WPNC47567.2019.8970184

Garello, R., Samson, J., Spirito, M., & Wymeersch, H. (2012). Peer-to-peer cooperative positioning part II: Hybrid devices with GNSS & terrestrial ranging capability. *InsideGNSS*, *7*(4), 56–64. https://www.insidegnss.com/auto/julyaug12-WP.pdf

Gentry, C. (2009). Fully homomorphic encryption using ideal lattices. *Proc. of the 41st Annual ACM Symposium on Theory of Computing (STOC '09),* Bethesda, MD, 169–178. https://doi.org/10.1145/1536414.1536440

Gentry, C., Sahai, A., & Waters, B. (2013). Homomorphic encryption from learning with errors: Conceptually-simpler, asymptotically-faster, attribute-based. *Proc. of the 33rd Annual Cryptology Conference (CRYPTO 2013),* Santa Barbara, CA, 75–92. https://doi.org/10.1007/978-3-642-40041-4_5

Gogoi, N., Minetto, A., & Dovis, F. (2019). On the cooperative ranging between android smartphones sharing raw GNSS measurements. *Proc. of the 2019 IEEE 90th Vehicular Technology Conference (VTC2019-Fall),* Honolulu, HI, 1–5. https://doi.org/10.1109/VTCFall.2019.8891320

Harvey, D. (2014). Faster arithmetic for number-theoretic transforms. *Journal of Symbolic Computation*, *60*, 113–119. https://doi.org/10.48550/arXiv.1205.2926

Hernandez, G., LaMountain, G., & Closas, P. (2020). Privacy-preserving cooperative positioning. *Proc. of the 33rd International Technical Meeting of the Satellite Division of the Institute of Navigation (ION GNSS+ 2020),* 2667–2675. https://doi.org/10.33012/2020.17531

Hernandez, G., LaMountain, G., & Closas, P. (2023). Proximity-based positioning scheme with multi-layer privacy. *Proc. of the 2023 IEEE/ION Position, Location and Navigation Symposium (PLANS),* Monterey, CA, 235–242. https://doi.org/10.1109/PLANS53410.2023.10140021

Hofmann-Wellenhof, B., Lichtenegger, H., &Wasle, E. (2007). *GNSS–global navigation satellite systems: GPS, GLONASS, Galileo, and more*. Springer Science & Business Media. https://doi.org/10.1007/978-3-211-73017-1

Holcer, S., Torres-Sospedra, J., Gould, M., & Remolar, I. (2020). Privacy in indoor positioning systems: A systematic review. *Proc. of the 2020 International Conference on Localization and GNSS (ICL-GNSS),* Tampere, Finland, 1–6. https://doi.org/10.1109/ICL-GNSS49876.2020.9115496

Huang, B., Yao, Z., Cui, X., & Lu, M. (2015). Dilution of precision analysis for GNSS collaborative positioning. *IEEE Transactions on Vehicular Technology*, *65*(5), 3401–3415. https://doi.org/10.1109/TVT.2015.2436700

Huang, B., Yao, Z., Cui, X., Lu, M., & Guo, J. (2014). GNSS collaborative positioning and performance analysis. *Proc. of the 27th International Technical Meeting of the Satellite Division of the Institute of Navigation (ION GNSS+ 2014)*, Tampa, Florida, 1920–1930. https://www.ion.org/publications/abstract.cfm?articleID=12384

Ibarrondo, A., & Viand, A. (2021). Pyfhel: Python for homomorphic encryption libraries. *Proc. of the 9th on Workshop on Encrypted Computing & Applied Homomorphic Cryptography (WAHC '21),* New York, NY, 11–16. https://doi.org/10.1145/3474366.3486923

Kaplan, E. D. (Ed.). (2006). *Understanding GPS: Principles and applications* (2nd ed.). Artech House.

Kassas, Z. M., Closas, P., & Gross, J. (2019). Navigation systems panel report navigation systems for autonomous and semiautonomous vehicles: Current trends and future challenges. *IEEE Aerospace and Electronic Systems Magazine*, *34*(5). https://doi.org/10.1109/MAES.2019.2906971

Katz, J., & Lindell, Y. (2007). *Introduction to modern cryptography: Principles and protocols*. Chapman and Hall/CRC.

Katz, J., & Lindell, Y. (2020). *Introduction to modern cryptography*. CRC press.

Laoudias, C., Moreira, A., Kim, S., Lee, S., Wirola, L., & Fischione, C. (2018). A survey of enabling technologies for network localization, tracking, and navigation. *IEEE Communications Surveys & Tutorials*, *20*(4), 3607–3644. https://doi.org/10.1109/COMST.2018.2855063

Lohan, E. S., Shubina, V., & Niculescu, D. (2022). Perturbed-location mechanism for increased user-location privacy in proximity detection and digital contact-tracing applications. *Sensors*, *22*(2), 687. https://doi.org/10.3390/s22020687

Lyubashevsky, V., & Micciancio, D. (2006). Generalized compact knapsacks are collision resistant. *Proc. of the 33rd International Colloquium on Automata, Languages, and Programming (ICALP 2006),* Venice, Italy, 144–155. https://doi.org/10.1007/1178700613

Lyubashevsky, V., Peikert, C., & Regev, O. (2010). On ideal lattices and learning with errors over rings. *Proc. of the Annual International Conference on the Theor and Applications of Cryptographic Techniques (EUROCRYPT 2010),* French Riviera, 1–23. https://doi.org/10.1007/978-3-642-13190-5_1

Minetto, A., Bello, M. C., & Dovis, F. (2022). DGNSS cooperative positioning in mobile smart devices: A proof of concept. *IEEE Transactions on Vehicular Technology*, *71*(4), 3480–3494. https://doi.org/10.1109/TVT.2022.3148538

Minetto, A., Cristodaro, C., & Dovis, F. (2017). A collaborative method for positioning based on GNSS inter agent range estimation. *Proc. of the 2017 25th European Signal Processing Conference (EUSIPCO),* Kos, Greece, 2714–2718. https://doi.org/10.23919/EUSIPCO.2017.8081704

Morton, Y. J., van Diggelen, F., Spilker Jr, J. J., Parkinson, B. W., Lo, S., & Gao, G. (2021). *Position, navigation, and timing technologies in the 21st century: Integrated satellite navigation, sensor systems, and civil applications*. John Wiley & Sons. https://doi.org/10.1002/9781119458449

Pany, T. (2010). *Navigation signal processing for GNSS software receivers*. Artech House.

Ranganathan, A., & Capkun, S. (2017). Are we really close? Verifying proximity in wireless systems. *IEEE Security & Privacy*, *15*(3), 52–58. https://doi.org/10.1109/MSP.2017.56

Sathaye, H., Schepers, D., Ranganathan, A., & Noubir, G. (2019). Wireless attacks on aircraft instrument landing systems. *Proc. of the 28th USENIX Security Symposium (USENIX Security 19),* Santa Clara, CA, 357–372. https://www.usenix.org/conference/usenixsecurity19/presentation/sathaye

Tahir, M., Afzal, S. S., Chughtai, M. S., & Ali, K. (2018). On the accuracy of inter-vehicular range measurements using GNSS observables in a cooperative framework. *IEEE Transactions on Intelligent Transportation Systems*, *20*(2), 682–691. https://doi.org/10.1109/TITS.2018.2833438

van Dijk, M., Gentry, C., Halevi, S., & Vaikuntanathan, V. (2010). Fully homomorphic encryption over the integers. *Proc. of the Annual International Conference on The Theory and Applications of Cryptographic Techniques (EUROCRYPT 2010),* French Riviera, 24–43. https://doi.org/10.1007/978-3-642-13190-5_2

Williams, N., Darian, P. B., Wu, G., Closas, P., & Barth, M. (2022). Impact of positioning uncertainty on connected and automated vehicle applications. *SAE International Journal of Connected and Automated Vehicles*, *6*(2). https://doi.org/10.4271/12-06-02-0010

Wu, P., Calatrava, H., Imbiriba, T., & Closas, P. (2023). Jammer classification with federated learning. *Proc. of the 2023 IEEE/ION Position, Location and Navigation Symposium (PLANS),* Monterey, CA, 228–234. https://doi.org/10.1109/PLANS53410.2023.10140124

How to cite this article: Hernandez, G., LaMountain, G., & Closas, P. (2023). Privacy-preserving cooperative GNSS positioning. *NAVIGATION, 70*(4). https://doi.org/10.33012/navi.625

APPENDIX

A | ADDITIONAL DETAILS ON HOMOMORPHIC ENCRYPTION

This appendix provides complementary material to expand upon the basics discussed in Section 3. The discussion begins with a definition of the ring LWE (RLWE)

problem and its implementation, followed by a commentary on some of the relevant parameters. Finally, some notes on the homomorphic properties under addition and multiplication of ciphertexts are given.

Definition 1 (Decision-RLWE (Fan & Vercauteren, 2012)). *Given a security parameter* λ, *let* $f(x)$ *be a cyclotomic polynomial* $\Phi_m(x)$ *with* $\deg(f)=\varphi(m)$ *depending on* λ *and set* $R=\mathbb{Z}[x]/f(x)$. *Let* $q=q(\lambda)\geq 2$ *be an integer. For a random element* $\mathbf{s}\in R_q$ *and a distribution* $\chi=\chi(\lambda)$ *over R,* $\mathbf{A}_{\mathbf{s},\chi}^{(q)}$ *denotes the distribution obtained by choosing a uniformly random element* $\mathbf{a}\leftarrow R_q$ *and a noise term* $\mathbf{e}\leftarrow\chi$ *and outputting* $(\mathbf{a},[\mathbf{a}\cdot\mathbf{s}+\mathbf{e}]_q)$. *The decision-RLWE*$_{d,q,\chi}$ *problem is to distinguish between the distribution* $\mathbf{A}_{\mathbf{s},\chi}^{(q)}$ *and the uniform distribution* $R_q^2\leftarrow\mathcal{U}$

The polynomial ring $R=\mathbb{Z}[x]/(f(x))$, where $f(x)\in\mathbb{Z}[x]$ is a monic irreducible polynomial of degree d. A popular choice for $f(x)$ is x^d+1, a cyclotomic, where $d=2^n$. This implies that the random variables are sampled as random polynomials with an indicated degree, defined by $\deg(f)$, where the m-th cyclotomic number field is considered as $\varphi(m)$. Additionally, the random element $\mathbf{a}$ is uniformly distributed from the set of polynomials within the ring R that contain coefficients in $\mathbb{Z}_q$. Let $\mathbb{Z}_q$ denote a set of integers $(-q/2,q/2]$, where $q>1$, and R_q denote the set of polynomials in R with coefficients in $\mathbb{Z}_q$. According to Lyubashevsky et al. (2010), the shape of q is independent of the hardness of the problem, and $\mathbf{s}$ can be distributed from R_2 as a polynomial of degree n. With ring R, the expansion factor limits the exponential coefficient growth and is defined as $\gamma_R=max\{\|\mathbf{a}\cdot\mathbf{b}\|/(\|\mathbf{a}\|\cdot\|\mathbf{b}\|):\mathbf{a},\mathbf{b}\in R\}$ (Fan & Vercauteren, 2012; Lyubashevsky & Micciancio, 2006). Furthermore, because the noise is distributed according to χ, it is considered to be B-bounded if the support of χ is in $[-B,B]$ (Brakerski, 2012; Fan & Vercauteren, 2012).

Definition 1 contains the ingredients needed to create the four main functions that define the BFV scheme, namely, SecretKeyGen, PublicKeyGen, Encryption, and Decryption. Note that the scheme itself does not include any homomorphic operations; instead, its main focus is on how to encrypt and decrypt a message. Additionally, these schemes depend on common parameters, denoted as θ hereafter, which include the security parameter (λ), polynomial degree (n), modulus value (t) of the plaintext coefficient, and modulus value (q) of the ciphertext coefficient, which are dependent on one another. Later, the homomorphic properties of this cryptographic scheme will be analyzed under certain operations such addition and multiplication. In the SecretKeyGen and PublicKeyGen algorithms, a pair of private and public keys is created, respectively. These keys are required if the (n-th) user desires to encrypt and decrypt data. The public key is kept with the user that created the keys (n-th user) and is also distributed to any (m-th) user that wishes to communicate with the user that created the keys. The Encryption algorithm takes in as an argument this public key and the message and uses the public key to encrypt a message. The Decryption algorithm uses the private key to decrypt a message. A brief description of these functions, as detailed in Brakerski (2012); Fan & Vercauteren (2012), is provided here for the sake of completeness.

Encryption Scheme

- SecretKeyGen (θ): The secret key is sampled from χ. The secret key is then set as $\mathbf{Sk}=\mathbf{s}\leftarrow\chi$.
- PublicKeyGen (θ, $\mathbf{Sk}$): $\mathbf{a}$ is uniformly sampled from R_q, and $\mathbf{e}$ is sampled from the χ distribution. The public key is set as follows:

$$\mathbf{Pk}=([-(\mathbf{a}\cdot\mathbf{s}+\mathbf{e})]_q,\mathbf{a}) \tag{A1}$$

- Encryption ($\mathbf{Pk}$, $\mathbf{m}$, θ): Let $\mathbf{m} \in R_t$ be the message to be encrypted, $\mathbf{p}_0 = \mathbf{Pk}[0] = [-(\mathbf{a} \cdot \mathbf{s} + \mathbf{e})]_q$, and $\mathbf{p}_1 = \mathbf{Pk}[1] = \mathbf{a}$. $\mathbf{e}_1$, $\mathbf{e}_2 \leftarrow \chi$ and $\mathbf{u} \leftarrow R_2$ are sampled. The ciphertext is generated as follows:

$$\mathrm{ct} = \left([\mathbf{p}_0 \cdot \mathbf{u} + \mathbf{e}_1 + \Delta \cdot \mathbf{m}]_q, [\mathbf{p}_1 \cdot \mathbf{u} + \mathbf{e}_2] \right) \tag{A2}$$

- Decryption ($\mathbf{Sk}$, ct, θ): Let $\mathbf{s} = \mathbf{Sk}$, $\mathbf{c}_0 = \mathrm{ct}[0] = [\mathbf{p}_0 \cdot \mathbf{u} + \mathbf{e}_1 + \Delta \cdot \mathbf{m}]_q$, and $\mathbf{c}_1 = \mathrm{ct}[1] = \mathbf{p}_1 \cdot \mathbf{u} + \mathbf{e}_2$. The ciphertext is decrypted as follows:

$$\mathbf{m} = \left[\left\lfloor \frac{t \cdot [\mathbf{c}_0 + \mathbf{c}_1 \cdot \mathbf{s}]_q}{q} \right\rceil \right]_t \tag{A3}$$

From the SecretKeyGen algorithm, the secret key ($\mathbf{Sk}$) is created from the χ distribution, the same distribution used to generate the noise vector ($\mathbf{e}$). This also indicates that the secret key polynomial coefficient values are within the domain of $\{-1, 0, 1\}$. As the size of the secret key (which depends on the polynomial degree parameter n) increases, it becomes more difficult for other users to recreate the secret key. Later, the impact of a large n for the remainder of the cryptosystem will be explored.

As shown in the encryption process, the public key is required in order to encrypt a message. The message itself must reside within the domain of the coefficient of the plaintext, indicating that $\mathbf{m} \in R_t$. Increasing the value of t will increase the plaintext space. With the public key, the additional noise polynomials, $\mathbf{u}$, $\mathbf{e}_1$, and $\mathbf{e}_2$, and the value $\Delta = \lfloor q/t \rfloor$, the message is converted to ciphertext. All of the additional polynomials contain coefficients that are small, similar to the error polynomial ($\mathbf{e}$) that exists within the public key. With the added noise, the ciphertext obtains an initial noise level, which is seen within the decryption process. To recover the message within the ciphertext, the ciphertext noise level must not become too high, as this would cause the message to be corrupted and not recovered. The noise threshold within the decryption process is discussed next.

The decryption process requires the secret key that was paired with the public key that encrypted the input ciphertext. With the secret key, the message is successfully decrypted if the noise level of the ciphertext remains below the noise threshold. Again, the ciphertext noise comes from the error vectors, $\mathbf{e}_1$ and $\mathbf{e}_2$, introduced in the Encryption algorithm. If Equation (A3) is expanded in terms of the public key $\mathbf{Pk}$ and if the noise variables are used, the total noise and upper noise bound within the ciphertext can be determined as follows:

$$\begin{aligned} \mathbf{c}_0 + \mathbf{c}_1 \cdot \mathbf{s} &= \mathbf{p}_0 \cdot \mathbf{u} + \mathbf{e}_1 + \Delta \cdot \mathbf{m} + \mathbf{p}_1 \cdot \mathbf{u} \cdot \mathbf{s} + \mathbf{e}_2 \cdot \mathbf{s} \mod q \\ &= -\mathbf{a} \cdot \mathbf{s} \cdot \mathbf{u} - \mathbf{e} \cdot \mathbf{u} + \mathbf{e}_1 + \Delta \cdot \mathbf{m} + \mathbf{a} \cdot \mathbf{u} \cdot \mathbf{s} + \mathbf{e}_2 \cdot \mathbf{s} \mod q \\ &= \Delta \cdot \mathbf{m} - \mathbf{e} \cdot \mathbf{u} + \mathbf{e}_1 + \mathbf{e}_2 \cdot \mathbf{s} \mod q \end{aligned} \tag{A4}$$

Because the ciphertexts, $\mathbf{c}_0$ and $\mathbf{c}_1$, consist of polynomial rings, the operations are polynomial addition and multiplication. From Equation (A4), the total noise is given by $\mathbf{v} = \mathbf{e} \cdot \mathbf{u} + \mathbf{e}_1 + \mathbf{e}_2 \cdot \mathbf{s}$. Because $\mathbf{e}, \mathbf{e}_1, \mathbf{e}_2 \leftarrow \chi$ and $\mathbf{u}, \mathbf{s} \leftarrow R_2$, the bound of the ciphertext noise is given by $\| \mathbf{v} \| \le B \cdot (2 \cdot \delta_R + 1)$. This upper bound is constructed from the expansion factor of the ring R and the upper bound from the χ distribution, and the error polynomial coefficients must satisfy the norm as $\| \mathbf{r} \| < 1$. Accounting for the noise and error terms, $\mathbf{c}_0 + \mathbf{c}_1 \cdot \mathbf{s}$ is set as $\Delta \cdot \mathbf{m} + \mathbf{v} + q \cdot \mathbf{r}$; by applying this expression in the decryption expression (Equation (A3)), the following is obtained:

$$\begin{aligned}\mathbf{m} &= \left[\left\lfloor \frac{t\cdot[\mathbf{c}_0+\mathbf{c}_1\cdot\mathbf{s}]_q}{q} \right\rceil\right]_t \\ &= \left[\left\lfloor \frac{t}{q}\cdot(\Delta\cdot\mathbf{m}+\mathbf{v}+q\cdot\mathbf{r}) \right\rceil\right]_t \\ &= \left[\left\lfloor \left(\frac{t}{q}\right)\Delta\cdot\mathbf{m}+\left(\frac{t}{q}\right)\mathbf{v}+t\cdot\mathbf{r}) \right\rceil\right]_t \\ &= \left[\mathbf{m}+\left\lfloor \frac{t}{q}(\mathbf{v}-\epsilon\cdot\mathbf{m}) \right\rceil+t\cdot\mathbf{r}\right]_t\end{aligned} \tag{A5}$$

Based on Equation (A5), where $\epsilon = q/t - \Delta < 1$, the decryption will be successful if the rounding term $(t/q)\cdot\|\mathbf{v}-\big((q/t)-\Delta\big)\cdot\mathbf{m}\| < 1/2$. The term $t\cdot\mathbf{r}$ vanishes with mod q, and lastly, $\mathbf{m}$ must exist within R_t. These results prove Lemma 1 in Brakerski (2012); Fan & Vercauteren (2012). Later, we will show how the ciphertext noise level increases with the number of homomorphic operations.

Encryption parameters. Based on the cryptosystem above, the following parameters have an impact on the size of the ciphertext or its noise level. The first parameter is the bit-length security parameter, λ. Roughly 2^λ computations are required for known attacks to determine the secret key; thus, as the security parameter increases, so does the computational cost for an attacker (Katz & Lindell, 2020). The second parameter is the modulus value q of the ciphertext coefficient, which creates a boundary of $(-q/2, q/2]$. As the boundary increases, the ciphertext domain increases, allowing for more possible ciphertext options to be mapped to and enabling the ciphertext to support more operations, which consequently increases the noise threshold. Increasing q does not signify any additional hardness within the problem (Brakerski, 2012; Fan & Vercauteren, 2012). The plaintext coefficient modulus, represented as the parameter t, creates the boundary $(-t/2, t/2]$ (Brakerski, 2012; Fan & Vercauteren, 2012). As the boundary increases, the plaintext polynomial also increases, and when noise is added, the noise level increases as well.

Homomorphic encryption under addition. When two ciphertexts are added, it can be observed that the messages within the ciphertexts are added as well, $\mathbf{m}_1+\mathbf{m}_2$. At the same time, the noise of each of these ciphertexts is also added, such that the overall noise is the sum of the noise contained in each ciphertext, which is shown as follows:

$$\begin{aligned}[\mathrm{ct}_3]_q &= [\mathbf{ct}_1+\mathbf{ct}_2]_q \\ &= ([\mathbf{p}_0\mathbf{u}_1+\mathbf{e}_{1,1}+\Delta\mathbf{m}_1]_q, [\mathbf{p}_1\mathbf{u}_1+\mathbf{e}_{2,1}]) + ([\mathbf{p}_0\mathbf{u}_2+\mathbf{e}_{1,2}+\Delta\mathbf{m}_2]_q, [\mathbf{p}_1\mathbf{u}_2+\mathbf{e}_{2,2}]) \\ &= \Delta[\mathbf{m}_1+\mathbf{m}_2]_t - \mathbf{e}\cdot(\mathbf{u}_1+\mathbf{u}_2) + (\mathbf{e}_{1,1}+\mathbf{e}_{1,2}) + (\mathbf{e}_{2,1}+\mathbf{e}_{2,2})\cdot\mathbf{s} \\ &= \Delta\cdot[\mathbf{m}_1+\mathbf{m}_2]_t + \mathbf{v}_1+\mathbf{v}_2 - \epsilon\cdot t\cdot\mathbf{r}\end{aligned}$$

From the above expression, let $\mathbf{m}_1+\mathbf{m}_2 = [\mathbf{m}_1+\mathbf{m}_2]_t + t\cdot\mathbf{r}$. The summation of two ciphertexts will give the same result as $\mathbf{m}_1+\mathbf{m}_2 \in \mathbb{R}_t$, with the noise also increasing. For the noise metric, the noise will increase by 1 bit.

Homomorphic encryption under multiplication. The product of two ciphertexts experiences a faster noise growth than the addition operation case. Lemma 2 in Fan & Vercauteren (2012) and Brakerski (2012) provides an in-depth analysis of the multiplication operation. While skipping the technical details here for the sake of clarity, it is important to emphasize that the noise will increase, as observed for the addition operation, but will increase by multiple bits (Brakerski, 2012).