Privacy-preserving cooperative positioning

Guillermo Hernandez, *Electrical and Computer Engineering Dept., Northeastern University*, Boston, MA, USA.
Gerald LaMountain, *Electrical and Computer Engineering Dept., Northeastern University*, Boston, MA, USA.
Pau Closas, *Electrical and Computer Engineering Dept., Northeastern University*, Boston, MA, USA.

BIOGRAPHIES

Guillermo Hernandez is a PhD candidate in the Department of Electrical and Computer Engineering at Northeastern University, Boston, MA. His research interests include signal processing, applied cryptography, and GNSS positioning.

Gerald LaMountain is a PhD candidate and Dean's Fellow in the Department of Electrical and Computer Engineering at Northeastern University, Boston, MA. His current research involves building upon traditional probabilistic techniques for dynamic state estimation and applying those techniques to problems with applications in positioning and localization.

Pau Closas is Assistant Professor at Northeastern University, Boston, MA. He received the MS and PhD degrees in Electrical Engineering from UPC in 2003 and 2009. He also holds a MS in Advanced Mathematics from UPC, 2014. His primary areas of interest include statistical signal processing, robust stochastic filtering, and machine learning, with applications to positioning systems and wireless communications. He is the recipient of the 2014 EURASIP Best PhD Thesis Award, the 9th Duran Farell Award, the 2016 ION Early Achievements Award, and a 2019 NSF CAREER Award.

ABSTRACT

We address the issue of user privacy in the context of "collaborative" positioning, wherein information is passed between and processed by multiple cooperative agents with the goal of achieving high levels of positioning accuracy. In particular, we evaluate the feasibility of applying a layer of encryption to a linear least squares (LS) algorithm for providing position, velocity, and time (PVT) estimates to a user based on information exchanged between neighboring receivers. The goal of such a scheme is to facilitate the requisite transfer and processing of GNSS measurements to achieve improved performance over single-user positioning, as demonstrated in the literature, for instance in mitigating errors induced by ionospheric propagation. Additionally, we wish to maintain that other agents or outside observers do not have access to the information required to locate a given user. We accomplish this by employing "homomorphic encryption" methodologies, in which fundamental mathematical operations may be performed on encrypted data to produce encrypted outputs, without providing access to either input or output to the agent performing the operation. The performance of the LS collaborative positioning methodology is evaluated both with and without encryption, and the results compared to each other as well as to the output of "single-user positioning" where information from other receivers is not used to mitigate the effects of atmospheric interferences. We show that the application of cooperative methodology results in an increase in performance under interference conditions, as compared to the single-user approach, and additionally that the application of homomorphic encryption to this LS approach yields little or no loss in estimation performance over the traditional, unencrypted cooperative LS algorithm.

INTRODUCTION

Among the ever expanding toolbox of methods for providing positioning in modern technology, few are as continually prevalent as Global Navigation Satellite System (GNSS) [11]. As the research community in GNSS continues to expand and investigate innovative ideas to increase the availability, precision and reliability of these systems [1, 3, 4], one technique that has been gaining traction is so-called "collaborative" positioning [5, 9, 10, 12]. In collaborative

positioning, information is passed between and processed by multiple agents, or collaborators, with the goal of minimizing interference and achieving high levels of positioning accuracy for each or some of the collaborators. One issue which may arise under such a scheme is that of user privacy [2]. Although these techniques may be desirable for the precision which they might provide, it may not always be possible or desirable to expose information which might be used to identify a user's location to collaborators. Of research interest, then, is 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.

One approach to the task of securing user information is by employing encryption: encrypted data can be broadcast to one or more collaborators without compromising the details of the message, thus preventing any collaborator or outside observer from obtaining information about another user. Of particular interest to the application of encrypted collaborative positioning are so-called "homomorphic" encryption schemes. Homomorphic is a term used to describe encryption schemes wherein mathematical operations may be performed on encrypted data, or cyphertext, 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 are possible in a way that doesn't expose the details of the operation, either at the input or the output, to the agent performing the manipulations. Each agent in the encryption, then, can contribute the information that they need to the scheme, and receive the benefits of cooperative positioning, without exposing the details of their own position to any other user.

Homomorphic encryption schemes fall broadly into three categories: Partially Homomorphic Encryption (PH), 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 a loss of computational fidelity occurs. Namely, a Partially Homomorphic Encryption (PHE) scheme limits an encryption scheme to only one operation performed over ciphertexts; addition or multiplication. A Somewhat Homomorphic Encryption (SWHE) scheme, by contrast, limits the number of operations performed, but allows both addition and multiplication to be performed over ciphertexts. Finally, in many ways combining the best of both PHE and SWHE, are Fully Homomorphic Encryption (FHE) schemes which can be used to perform an unlimited number of both addition and multiplication operations over ciphertexts. Research on the topic of FHE schemes has expanded immensely since the first functional scheme introduced by Gentry [6]. FHE in general are computational expensive and complex during a multiplication operation. Within the FHE scheme category we looked at a Learning with Error (LWE)-based encryption Approximate Eigenvector Method scheme [7]. This scheme addresses the computation complexity for a multiplication operation performed over ciphertexts. In addition, previous FHE schemes, users have an additional evaluation key, where an evaluator required those keys to perform any operation. The Approximate Eigenvector scheme eliminated the existence of any evaluation key which allows third-party device to act as an evaluator to directly perform the desired operations to the ciphertexts.

In this contribution, we evaluate the feasibility of applying a layer of homomorphic encryption to a linear least squares (LS) algorithm for providing position, velocity, and time (PVT) estimates to a user, based on information exchange from neighboring receivers. This methodology will leverage a single pseudorange difference that cancels the effects of the certain impairments (i.e. tropospheric delay and ionospheric delay) between two receivers [8]. Each user will then independently compute each other's positions in an encrypted domain, before returning the result of this computation to the other user for decryption and fusion to determine the final positioning result. The required computations are performed locally at each receiver, leveraging others' data, and as such, with respect to user privacy, the variables involved in the LS should be encrypted (and thus not exposed) to the users responsible for computing the PVT solution. Using the homomorphic encryption methodology in a least square problem, in this case the PVT computation, the results are expected to be the same compared to the least square problem with no encryption scheme in place. The results will establish a concrete understanding that having a secure protocol will not have any effect in terms of performance degradation while it adds an important layer of privacy to the user accessing the service. Parallel to this idea, the problem with no encryption will serve as a base to compare the results to the case where the homomorphic encryption was implemented.

This paper is categorized into three main sections. The first section details a Cooperative Positioning (CoPo) scheme and implementation that preserves privacy between collaborating peers. A second section provides some technical background on Fully Homomorphic Encryption, which is required in order to implement the Private CoPo scheme. This is followed by the third section where some results are discussed to validate the proposed solution.

COOPERATIVE POSITIONING

We describe a positioning algorithm where two arbitrarily close receivers exchange GNSS observables (code phase in this article) such that they can remove atmospheric errors by combining those measurements. Under such a scheme pseudoranges measured by two receivers (n and m) are substracted, in the vein of standard differential GNSS (DGNSS) schemes, with the particularity that none of those receivers has accurate knowledge of its position and is benefiting from the process (i.e. no base station is present). The rest of the paper leverages this algorithm to propose a scheme that preserves privacy among users, that is, the CoPo can be implemented without the n-th receiver being able to infer the position of the m-th receiver, and vice versa.

Differential GNSS as a cooperative positioning scheme

A GNSS receiver processes signals to compute the so-called observables, namely pseudorange and carrier-phase measurements to L visible satellites. These observables are related to position, velocity, and time (PVT) unknown quantities of the receiver, so they are typically used to solve for those. Let's assume that two nearby receivers (n and m) are able to compute those observations, the pseudorange observable for the n-th is modeled as

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

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 $i = \{1, \dots, L\}$; $\delta t^{(n)}$ 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 delay errors due to the troposphere and the ionosphere, 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 $\mathbf{R}^{(n)} = \mathrm{diag}(\sigma_{n,1}^2, \dots, \sigma_{n,L}^2)$. In vector form, the L pseudoranges can be gathered in

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

Through differencing pseudoranges from two neighboring receivers, a new observable can be obtained that is free of the common tropospheric and ionospheric associated delays. That is,

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

does not have the contributions of T_i and I_i. The random term is then $\eta_i^{(n,m)} = \epsilon_i^{(n)} - \epsilon_i^{(m)} \sim \mathcal{N}(0, \sigma_{n,i}^2 + \sigma_{m,i}^2)$, whose variance is clearly increased as a consequence of the processing. In vector form:

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

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

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

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

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

and one can take a first order Taylor expansion to linearize the problem at some arbitrary points $\boldsymbol{\mu}_0^{(n)}$ and $\boldsymbol{\mu}_0^{(m)}$:

$$\mathbf{h}(\boldsymbol{x}^{(n)}, \boldsymbol{x}^{(m)}) \approx \mathbf{h}(\boldsymbol{\mu}_0^{(n)}, \boldsymbol{\mu}_0^{(m)}) + \nabla_{\boldsymbol{x}}\mathbf{h}(\boldsymbol{\mu}_0^{(n)}, \boldsymbol{\mu}_0^{(m)})(\boldsymbol{x} - \boldsymbol{\mu}_0) \tag{7}$$

where $\boldsymbol{x}$ 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 $\boldsymbol{H} = [\boldsymbol{H}^{(n)}, -\boldsymbol{H}^{(m)}]$, with

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

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

Then, once the problem in (5) is linearized, it can be easily solve analytically through an iterative least squares procedure. To do so, we rearrange the term as

$$\mathbf{y}^{(n,m)} = \Delta\boldsymbol{\rho}^{(n,m)} + \boldsymbol{H}\boldsymbol{\mu}_0 - \mathbf{h}(\boldsymbol{\mu}_0^{(n)}, \boldsymbol{\mu}_0^{(m)}) \tag{9}$$

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

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

$$= \boldsymbol{H}^{(n)} \begin{bmatrix} \boldsymbol{p}^{(n)} \\ c\delta t^{(n)} \end{bmatrix} - \boldsymbol{H}^{(m)} \begin{bmatrix} \boldsymbol{p}^{(m)} \\ c\delta t^{(m)} \end{bmatrix} + \boldsymbol{\eta}\ , \tag{12}$$

where $\boldsymbol{H} = [\boldsymbol{H}^{(n)}, -\boldsymbol{H}^{(m)}]$ can be splitted in the terms associated to each receiver. Solving for $\hat{\boldsymbol{x}}^{(m)}$ in (12) as a least squares (LS) problem, and assuming that the n-th receiver has an estimate of $\hat{\boldsymbol{x}}^{(n)}$, a LS estimate for the m-th receiver can be readily computed as

$$\hat{\boldsymbol{x}}^{(m)} = ({\boldsymbol{H}^{(m)}}^\top \boldsymbol{H}^{(m)})^{-1} {\boldsymbol{H}^{(m)}}^\top (\boldsymbol{H}^{(n)}\hat{\boldsymbol{x}}^{(n)} - \mathbf{y}^{(n,m)}) \tag{13}$$

$$= \boldsymbol{\mu}_0^{(m)} - ({\boldsymbol{H}^{(m)}}^\top \boldsymbol{H}^{(m)})^{-1} {\boldsymbol{H}^{(m)}}^\top (\mathbf{y}^{(n,m)} - \boldsymbol{H}^{(n)}\hat{\boldsymbol{x}}^{(n)} + \boldsymbol{H}^{(m)}\boldsymbol{\mu}_0^{(m)})\ , \tag{14}$$

and similarly

$$\hat{\boldsymbol{x}}^{(n)} = ({\boldsymbol{H}^{(n)}}^\top \boldsymbol{H}^{(n)})^{-1} {\boldsymbol{H}^{(n)}}^\top (\mathbf{y}^{(n,m)} + \boldsymbol{H}^{(m)}\hat{\boldsymbol{x}}^{(m)}) \tag{15}$$

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

for the other unknown vector.

Privacy-aware cooperative positioning

We propose a "privacy-aware" modification to the CoPo scheme previously described, wherein each receiver can obtain an enhanced PVT solution after pseudorange combination by solving (13) and (15). The proposed scheme leverages a particular class of encryption methodologies that have homomorphic properties. That is, we consider crypto systems that allow certain operations (addition and multiplication) in the ciphertext without leaking information about the actual information.

The proposed scheme involves a key encryption method (an explanation of which is provided in the next section), where the n-th receiver generates a public and private key and encrypts its information to be transferred to the m-th user, along with the public key $\boldsymbol{Pk}^{(n)}$. Then m-th receiver is then in charge of solving part of (15) in an encrypted manner without leaking information about the n-th user, which is then only able to decipher the encrypted version of $\hat{\boldsymbol{x}}^{(n)}$. An analogous scheme would be required for computing (13) privately. It is important to note that the distribution of public key ($\boldsymbol{Pk}^{(n)}$) occurs once, as shown in Figure 1. There is no need to redistribute the public key; the public keys are required to return any desired data.

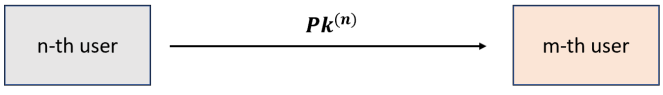

Figure 1: n-th user shares its own public key ($\boldsymbol{Pk}^{(n)}$)

Denoting the encryption operation with $\mathrm{Enc}(\cdot)$, which will be discussed in more detail in the next section, the scheme is such that the m-th receiver computes part of (15). In particular this expression can be rearranged as

$$\hat{\boldsymbol{x}}^{(n)} = \underbrace{\boldsymbol{A}^{(n)}\tilde{\mathbf{y}}^{(n)}}_{n\text{-th receiver}} - \boldsymbol{A}^{(n)} \underbrace{(\tilde{\mathbf{y}}^{(m)} - \boldsymbol{H}^{(m)}\hat{\boldsymbol{x}}^{(m)})}_{m\text{-th receiver}} \tag{17}$$

with

$$\mathbf{y}^{(n,m)} = \tilde{\mathbf{y}}^{(n)} - \tilde{\mathbf{y}}^{(m)} \tag{18}$$

$$\tilde{\mathbf{y}}^{(n)} = \boldsymbol{\rho}^{(n)} + \boldsymbol{H}^{(n)}\boldsymbol{\mu}_0^{(n)} - \mathbf{h}(\boldsymbol{\mu}_0^{(n)}) \tag{19}$$

$$\tilde{\mathbf{y}}^{(m)} = \boldsymbol{\rho}^{(m)} + \boldsymbol{H}^{(m)}\boldsymbol{\mu}_0^{(m)} - \mathbf{h}(\boldsymbol{\mu}_0^{(m)}) \tag{20}$$

$$\boldsymbol{A}^{(n)} = (\boldsymbol{H}^{(n)^\top}\boldsymbol{H}^{(n)})^{-1}\boldsymbol{H}^{(n)^\top} \tag{21}$$

where the two 4-dimensional vectors in (17) only carry information of the corresponding user. As a consequence, it is possible to implement an homomorphic encryption scheme such that decrypting

$$\text{Enc}(\boldsymbol{A}^{(n)} \cdot (\tilde{\mathbf{y}}^{(m)} - \boldsymbol{H}^{(m)}\hat{\boldsymbol{x}}^{(m)}))_{\boldsymbol{Pk}^{(n)}} = \text{Enc}(\boldsymbol{A}^{(n)})_{\boldsymbol{Pk}^{(n)}} \cdot \text{Enc}((\tilde{\mathbf{y}}^{(m)} - \boldsymbol{H}^{(m)}\hat{\boldsymbol{x}}^{(m)}))_{\boldsymbol{Pk}^{(n)}} \tag{22}$$

will actually solve for the receiver dependent calculations in (15). To that aim, the n-th user transfers the public key $\boldsymbol{Pk}^{(n)}$ for the m-th user to be able to encrypt its data as well as the other necessary

$$\mathcal{D}^{(n)} = \left\{\text{Enc}(\boldsymbol{A}^{(n)})\right\} \tag{23}$$

such that the operations (in the encrypted domain) at the m-th receiver involve a matrix/vector bit-wise product. As shown in Figure 2, the n-th user encrypts its own set of data (23), while the m-th user encrypts its own corresponding data using the n-th user's public key. Once the m-th user receives the encrypted data from the n-th user, it performs the encrypted calculations in the encrypted domain as shown in (22). The m-th user then returns the encrypted result to the n-th user, shown in Figure 3. When n-th user receives the encrypted results, using its private key $\boldsymbol{Sk}^{(n)}$, it decrypts the result and solves for (15). A similar manner is required to calculate (13). In this collaborative process, each user computes 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.

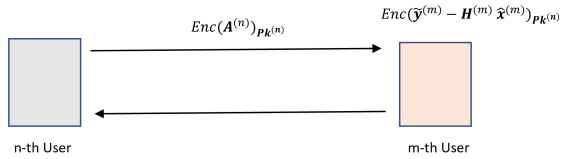

Figure 2: n-th and m-th users encrypted their corresponding data to calculate (22)

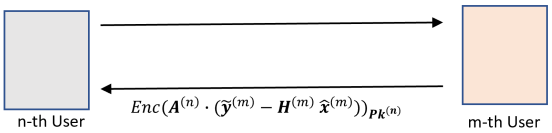

Figure 3: m-th user returns encrypted calculated results to the n-th user

FULLY HOMOMORPHIC ENCRYPTION SCHEME AND OPERATIONS

This section provides some technical background details on the cryptosystem considered in this work. Namely, a Fully Homomorphic Encryption (FHE) is considered, which shows homomorphism in addition and multiplication.

Homomorphic encryption is a term used to describe methods of data encryption that allow mathematical computation to be performed on encrypted data, such that the result of the computation, when decrypted, matches the result of the operations as though they had been performed without encryption. Typically, a homomorphic encryption 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 $\text{Enc}(\cdot)$ and decryption by $\text{Dec}(\cdot)$, and some operator $f(\cdot,\cdot)$, then [14] if homomorphic encryption is considered we have that $\text{Dec}(f(\text{Enc}(x), \text{Enc}(y))) = f(x, y)$ for two messages x and y. That is, the function can be applied to the encrypted data, then decrypted, and yield the same result as when no privacy is considered. The types of functions supported (typically multiplication and additions) depends on the considered cryptosystem [14], which is part of the topic in this study.

FHE Scheme background

The Eigenvector FHE scheme described herein provides security, by adding noise to message (μ). This noise is added using a public key and removed using a private key; both of these keys are created from the same source (user). The process of creating these keys are defined into three main sections: Key generation, Encryption and Decryption. Descriptions of each are explained below. Once these keys are created, the public key is distributed to a secondary, or any arbitrary number of devices where encrypted communication is desired. As an example, if Alice wants to communicate with Bob, Alice requires Bob's public key. This public key encrypts the message Alice sends to Bob. To read the message, Bob uses his secret key to decrypt the message.

The Eigenvector FHE key generator is divided into three parts: the **Setup**, **SecretKey**, and **PublicKey**. [7]

- **Setup**: Choose a modulus q of $\kappa = \kappa(\lambda, L)$ bits, lattice dimension parameter $n = n(\lambda, L)$, and error distribution $\chi = \chi(\lambda, L)$ appropriately for LWE that achieves at least 2^λ security against known attacks. Also, choose parameter $m = m(\lambda, L) = O(n\text{log}q)$. Let $params = (n, q, X, m)$. Let $\ell = \lfloor \text{log}q \rfloor + 1$ and $N = (n + \ell) \cdot l$.
- **SecretKey** ($params$): Sample $\boldsymbol{t}$, where $\boldsymbol{t} \in \mathbb{Z}_q^n$. Let $\boldsymbol{sk} = (1, -t_1, -t_2, \ldots, -t_n) \in \mathbb{Z}_q^{n+1}$.
  Let $\boldsymbol{v} = \text{Powersof}(\boldsymbol{sk})$.
- **PublicKey** ($params, \boldsymbol{sk}$): Generate a matrix $\boldsymbol{B} \in \mathbb{Z}_q^{m \times n}$ uniformly and an error vector $\boldsymbol{e} = \chi^m$. Set $\boldsymbol{b} = \boldsymbol{B} \cdot \boldsymbol{t} + \boldsymbol{e}$.
  Set $\boldsymbol{A}$ to be a $(n+1)-$column matrix, where $\boldsymbol{b}$ is the first column followed by $\boldsymbol{B}$. Public Key is $\boldsymbol{A}$

As seen above, the secret and public keys are randomly constructed; if a third-party device obtains a public key, it will be very difficult to obtain or determine the secret key to that specific public key. With the public key distributed to the public, Alice can encrypt a message using Bob's public key, without the worry of having a third-party user corrupt the message.

The process to encrypt a message, using the Eigenvetor FHE scheme requires only the public parameters ($params$), the public key, and the desired message (μ). Additionally, the Eigenvector FHE contains an encryption **Enc** function that encrypts μ. The following is a brief explanation of the the **Enc** function.[7]

- **Enc** ($params, \boldsymbol{Pk}, \mu$): To encrypt a message $\mu \in \mathbb{Z}_q$, sample a uniform matrix $\boldsymbol{R} \in \{0,1\}^{N \times m}$

$$\mathbf{C} = \text{Flatten}(\mu \cdot \mathbf{I}_N + \text{BitDecomp}(\mathbf{R} \cdot \mathbf{A})) \in \mathbb{Z}_q^{N \times N}$$

The **Enc** outputs the ciphertext($\boldsymbol{C}$), where the largest magnitude is 1. As stated before, one of the reasons this is secure contributes to the noise added to the public key, as we saw with the error vector ($\boldsymbol{e}$). The noise is incorporated through the *BitDecomp* function, which converts the values in the public key ($\boldsymbol{A}$) to bits. The *Flatten* function, ensures the magnitude of the ciphertext to be as most 1.

Lastly, the user with the secret key decrypts any incoming ciphertext through the use of its secret key and the decryption function (**Dec**). [7] describes two methods of decryption. One decryption method describes Regev's method seen in, [13]. The second decryption method, **MDec** function, is ideal when the modulo value (q) is a power of 2. [7]

- **MDec** ($params, \boldsymbol{Sk}, \boldsymbol{C}$): Observe that ($q = 2^{\ell-1}$) and the first $\ell - 1$ coefficients of $\boldsymbol{v}$ are $(1, 2, \ldots, 2^{\ell-2})$. Recover LSB($\mu$) from $\mu \cdot 2^{\ell-2} + small$, then recover the next LSB from $(\mu - \text{LSB}(\mu)) \cdot 2^{\ell-3} + small$, etc.

The decryption procedure decrypts the message if the error inserted by the error vector($\boldsymbol{e}$) remains small; $\boldsymbol{C} \cdot \boldsymbol{v} = \mu \cdot \boldsymbol{v} + small$. If the error is small, then the first $\ell - 1$ coefficients of $\boldsymbol{C} \cdot \boldsymbol{v}$ are $\mu \cdot \boldsymbol{g} + small$, where $\boldsymbol{g} = (1, 2, \ldots, 2^{\ell-2})$.

Properties

We can now analyze the FHE two operations performed on the ciphertext: addition and multiplication.

- To add ciphertexts $\boldsymbol{C}_1$, $\boldsymbol{C}_2 \in \mathbb{Z}_q^{n\times n}$

$$(\mathbf{C}_1 + \mathbf{C}_2)\cdot \mathbf{v} = \mu_1 \cdot \mathbf{v} + \mathbf{e}_1 + \mu_2 \cdot \mathbf{v} + \mathbf{e}_2 \tag{24}$$
$$= (\mu_1 + \mu_2)\cdot \mathbf{v} + (\mathbf{e}_1 + \mathbf{e}_2) \tag{25}$$

As send in (24), a ciphertext is $\mu \cdot \boldsymbol{v} + \boldsymbol{e}$. When adding two ciphertexts, the error value linearly increases. While the total error value remains small, the two cipher are directly added to one another.

- To multiply a ciphertext $\boldsymbol{C} \in \mathbb{Z}_q^{n\times n}$ by a know constant $\alpha \in \mathbb{Z}_q$. Let $\boldsymbol{M}_\alpha$ be the ciphertext of α. Observe:

$$\mathbf{M}_\alpha \cdot \mathbf{C} \cdot \mathbf{v} = \mathbf{M}_\alpha \cdot (\mu \cdot \mathbf{v} + \mathbf{e}) = \mu \cdot (\mathbf{M}_\alpha \cdot \mathbf{v}) + \mathbf{M}_\alpha \cdot \mathbf{e} \tag{26}$$
$$= \alpha \cdot \mu \cdot \mathbf{v} + \mathbf{M}_\alpha \cdot \mathbf{e} \tag{27}$$

Multiplying a ciphertext by a constant value (α) requires α to go through a similar process of encryption. The process requires α to convert its magnitude value to be at most 1. This is done by converting α into $\boldsymbol{M}_\alpha$, by setting $\boldsymbol{M}_\alpha$ equal to Flatten($\alpha \cdot \boldsymbol{I}_N$), where $\boldsymbol{I}_N$ is an identity matrix.

- To multiply two ciphertexts $\boldsymbol{C}_1$, $\boldsymbol{C}_2 \in \mathbb{Z}_q^{n\times n}$. Observe:

$$\mathbf{C}_1 \cdot \mathbf{C}_2 \cdot \mathbf{v} = \mathbf{C}_1 \cdot (\mu_2 \cdot \mathbf{v} + \mathbf{e}_2) + \mu_2 \cdot (\mu_1 \cdot \mathbf{v} + \mathbf{e}_1) + \mathbf{C}_1 \cdot \mathbf{e}_1 \tag{28}$$
$$= \mu_1 \cdot \mu_2 \cdot \mathbf{v} + (\mu_2 \cdot \mathbf{e}_1 + \mathbf{C}_1 \cdot \mathbf{e}_2) \tag{29}$$

Lastly, multiplying two ciphertexts together converts the error into $\mu_2 \cdot \boldsymbol{e}_1 + \boldsymbol{C}_1 \cdot \boldsymbol{e}_2$. This error will remain small, assuming that $\boldsymbol{C}_1$ and μ_2 are small. [7] provides more detail about setting tight bounds to ensure that the error remains minimal.

Homomorphic Encryption Approach for CoPo

The CoPo scheme described consists of the *Enc* and *MDec* functions and the two operations capable of being perform on the encryption layer. The CoPo scheme follows the same process of creating public and private keys from the *SecretKey* and *PublicKey* functions. as well as converted a value into ciphertext matrix. The CoPo prepares the encrypted package, send in (23). Note, any sent encrypted package is encrypted by the destination's public key. Once this set of variables arrives at the appropriate destination, the user encrypts its own information to perform the following operations in the encryption layer.

The first homomorphic operation addressed in the CoPo scheme is addition. For notation purposes, let

$$\boldsymbol{u} = \text{Enc}(\boldsymbol{A}^{(n)})$$
$$\boldsymbol{w} = \text{Enc}((\tilde{\mathbf{y}}^{(m)} - \boldsymbol{H}^{(m)}\hat{\boldsymbol{x}}^{(m)}))$$

Both of these vectors are composed of ciphertext matrices. where each matrix is of $(N \times N)$- dimensional and each matrix represents a binary value. The CoPo scheme algorithm adds each matrix as a element-by-element addition operation, in order to directly perform the homomorphic operation.

$$\boldsymbol{u}_{(\{1:b\},N,N,i)} + \boldsymbol{w}_{(\{1:b\},N,N,i)} = \boldsymbol{C}^u_{(\{1:b\},i)} + \boldsymbol{C}^w_{(\{1:b\},i)} \tag{30}$$

The multiplication homomorphic operation implemented in the CoPo, follows a similiar process from the additional operation. For notation purposes, let $\boldsymbol{B} = \text{Enc}(\boldsymbol{A}^{(n)})$ and let $\boldsymbol{z} = \text{Enc}(\tilde{\mathbf{y}}^{(m)} - \boldsymbol{H}^{(m)}\hat{\boldsymbol{x}}^{(m)})$. The scheme divides this operation to follow the multiplication format described in the previous section.

$$\boldsymbol{B}\cdot\boldsymbol{z} = \begin{bmatrix} \boldsymbol{B}_{(\{1:b\},N,N,1,1)} \cdot \boldsymbol{z}_{(\{1:b\},N,N,1)} + \boldsymbol{B}_{(\{1:b\},N,N,1,2)} \cdot \boldsymbol{z}_{\{1:b\},(N,N,2)} + \ldots + \boldsymbol{B}_{(\{1:b\},N,N,1,i)} \cdot \boldsymbol{z}_{(\{1:b\},N,N,i)} \\ \boldsymbol{B}_{(\{1:b\},N,N,2,1)} \cdot \boldsymbol{z}_{(\{1:b\},N,N,1)} + \boldsymbol{B}_{(\{1:b\},N,N,2,2)} \cdot \boldsymbol{z}_{\{1:b\},(N,N,2)} + \ldots + \boldsymbol{B}_{(\{1:b\},N,N,2,i)} \cdot \boldsymbol{z}_{(\{1:b\},N,N,i)} \\ \vdots \\ \boldsymbol{B}_{(\{1:b\},N,N,j,1)} \cdot \boldsymbol{z}_{(\{1:b\},N,N,1)} + \boldsymbol{B}_{(\{1:b\},N,N,j,2)} \cdot \boldsymbol{z}_{(\{1:b\},N,N,2)} + \ldots + \boldsymbol{B}_{(\{1:b\},N,N,j,i)} \cdot \boldsymbol{z}_{(\{1:b\},N,N,i)} \end{bmatrix} \tag{31}$$

This is achieved by having element-by-element multiplication. Same as before, since each element in $\text{Enc}(\boldsymbol{A}^{(n)})$ and the outcome of $\text{Enc}(\tilde{\boldsymbol{y}}^{(m)} - \boldsymbol{H}^{(m)}\hat{\boldsymbol{x}}^{(m)})$ are a ciphertext of $(N \times N)$- dimension; this element-by-element multiplication converts to matrix-by-matrix multiplication.

$$\boldsymbol{B}_{(\{1:b\},N,N,1,1)} \cdot \boldsymbol{z}_{(\{1:b\},N,N,1)} = \boldsymbol{C}_{\boldsymbol{B}_{(\{1:b\},1,1)}} \cdot \boldsymbol{C}_{\boldsymbol{z}_{(\{1:b\},1)}} \tag{32}$$

SIMULATION RESULTS

The proposed CoPo scheme was validate through simulated data. Particularly, two static and cooperative receivers (separated 1200 meters apart) were simulated in open-sky conditions. A Single Point Positioning (SPP) solution, (i.e. no cooperation) was obtained as benchmark without any ionospheric error and a second SPP solution was with ionospheric delay was also included; this ionospheric delay was modeled after Klobuchar's ionospheric model. Those were compared to the proposed CoPo scheme with and without encryption.

Figure 4 shows the results of the RMSE (root mean square error) for each scheme; SPP solutions, CoPo scheme, and encrypted CoPo scheme, each with and without ionospheric delay, at different tracking error variance values. As the tracking error variance value is relatively small, the ionospheric delay increases the RMSE value for the benchmark. In the both CoPo schemes (encrypted and non-encrypted), the ionospheric delay impact to the RMSE is almost negligible, due to the increase of the ionospheric delay variance between the two receivers and they both have similar results to that of the SPP solution without any ionospheric delay. When the tracking error variance value increases, the ionospheric delay is overlapped by the tracking error causing it to have no effect in the RMSE, yet the results for all the schemes are similar. It is also important to note, the integer bit value chosen effects the RMSE as well. At a smaller integer bit value, the RMSE for the CoPo (encrypted and non-encrypted) schemes, approaches the RMSE value for the benchmark. This occurs since information is lost when the integer bit value decreases.

RMSE				
Tracking Error Variance(m)	0.5	1	5	10
LS Solution (No Iono Delay) (m)	2.1703	4.5888	21.914	44.059
LS Solution (Iono Delay) (m)	4.0902	5.7442	21.849	44.208
CoPo Solution (No Iono Delay) (m)	2.1703	4.5887	21.915	44.059
CoPo Solution (Iono Delay) (m)	2.1703	4.5888	21.914	44.059
Encrypted CoPo Solution (No Iono Delay) (m)	2.1703	4.5887	21.915	44.059
Encrypted CoPo Solution (Iono Delay) (m)	2.1703	4.5888	21.914	44.059

Figure 4: RMSE Values at different tracking error variance

In addition to the RMSE values, the cost of implementing each schemes is significantly different, where the encrypted CoPo with and witout ionospheric delay is much larger than the other schemes. Figure 5 displays the number of operation during the convergence self-assessment per iteration. In the case for the CoPo scheme, (13) and (15) require one operation each, while in the case for the encrypted CoPo scheme, (22) for the n-th receiver and similar to the m-th receiver required a bit-wise multiplication operation, as shown in (31). In the worse-case $O(spb^2)$, where b is the bit size, s is the number of satellites, and p are the unknowns parameters of the user (position and clock offset).

Number of Operations Per Iteration	
LS Solution (No Iono Delay)	1
LS Solution (Iono Delay)	1
CoPo Solution (No Iono Delay)	2
CoPo Solution (Iono Delay)	2
Encrypted CoPo Solution (No Iono Delay)	4608
Encrypted CoPo Solution (Iono Delay)	4608

Figure 5: Scheme operational cost per scheme

CONCLUSIONS

This paper proposed a cooperative positioning (CoPo) scheme that is inspired in Differential GNSS (DGNSS) schemes, with the peculiarity that none of those collaborative receivers has exact knowledge of its location. The scheme assumes that those receivers can exchange observables (pseudoranges in this work) which are then combined in order to mitigate the effects of atmospheric errors that are common between closeby receivers. Additionally, the article proposes the use of advanced crypto-schemes in order to compute the operations involved in the CoPo scheme in a privacy-preserving manner. That is, in a way that no information regarding the position of the collaborative receiver is leaked to the other user benefiting from it. The scheme leverages the so-called Fully Homomorphic Encryption, allowing to perform certain operations like addition/multiplication in the encrypted messages. The article validates the proposed privacy-preserving CoPo scheme through a set of simulation results. To the best of the authors' knowledge, this is the first time a GNSS CoPo scheme is presented that features privacy-preserving properties.

ACKNOWLEDGEMENTS

This work was partially supported by the National Science Foundation under Awards CNS-1815349 and ECCS-1845833.

REFERENCES

[1] M. G. Amin, P. Closas, A. Broumandan, and J. L. Volakis. Vulnerabilities, threats, and authentication in satellite-based navigation systems [scanning the issue]. *Proceedings of the IEEE*, 104(6):1169–1173, 2016.

[2] L. Chen, S. Thombre, K. Järvinen, E. S. Lohan, A. Alén-Savikko, H. Leppäkoski, M. Z. H. Bhuiyan, S. Bu-Pasha, G. N. Ferrara, S. Honkala, et al. Robustness, security and privacy in location-based services for future IoT: A survey. *IEEE Access*, 5:8956–8977, 2017.

[3] D. Dardari, P. Closas, and P. M. Djurić. Indoor tracking: Theory, methods, and technologies. *IEEE Transactions on Vehicular Technology*, 64(4):1263–1278, 2015.

[4] D. Dardari, E. Falletti, and M. Luise. *Satellite and terrestrial radio positioning techniques: a signal processing perspective*. Academic Press, 2011.

[5] R. Garello, J. Samson, M. Spirito, and H. Wymeersch. Peer-to-peer cooperative positioning. *Inside Gnss*, 2012.

[6] C. Gentry. Fully homomorphic encryption using ideal lattices. In *Proceedings of the forty-first annual ACM symposium on Theory of computing*, pages 169–178, 2009.

[7] C. Gentry, A. Sahai, and B. Waters. Homomorphic encryption from learning with errors: Conceptually-simpler, asymptotically-faster, attribute-based. In *Annual Cryptology Conference*, pages 75–92. Springer, 2013.

[8] N. Gogoi, A. Minetto, and F. Dovis. On the Cooperative Ranging between Android Smartphones Sharing Raw GNSS Measurements. In *2019 IEEE 90th Vehicular Technology Conference (VTC2019-Fall)*, pages 1–5, 2019.

[9] B. Huang, Z. Yao, X. Cui, and M. Lu. Dilution of precision analysis for GNSS collaborative positioning. *IEEE Transactions on Vehicular Technology*, 65(5):3401–3415, 2015.

[10] B. Huang, Z. Yao, X. Cui, M. Lu, and J. Guo. GNSS collaborative positioning and performance analysis. In *ION GNSS*, pages 1920–1930, 2014.

[11] E. D. Kaplan, editor. *Understanding GPS: principles and applications*. Artech House, 2nd edition, 2006.

[12] A. Minetto, C. Cristodaro, and F. Dovis. A collaborative method for positioning based on GNSS inter agent range estimation. In *2017 25th European Signal Processing Conference (EUSIPCO)*, pages 2714–2718. IEEE, 2017.

[13] O. Regev. On lattices, learning with errors, random linear codes, and cryptography. *Journal of the ACM (JACM)*, 56(6):1–40, 2009.

[14] M. Van Dijk, C. Gentry, S. Halevi, and V. Vaikuntanathan. Fully homomorphic encryption over the integers. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 24–43. Springer, 2010.