Dissertation for the Faculty of Computer Science (FIN),
Otto von Guericke University (OVGU), Magdeburg

# Data Confidentiality for Distributed Sensor Fusion

Marko Ristic

October 5, 2022

Reviewers:

Prof. Dr.-Ing. Benjamin Noack (OVGU, Magdeburg, Germany)

...

# Contents

# Acknowledgements

Acks go here.

# Abstract

Distributed sensing and fusion algorithms are increasingly present in public computing networks and have led to a natural concern for data security in these environments. This thesis aims to present generalisable data fusion algorithms that simultaneously provide strict cryptographic guarantees on user data confidentiality. While fusion algorithms providing some degrees of security guarantees exist, these are typically either provided at the cost of solution generality or lack formal security proofs. Here, novel cryptographic constructs and state-of-the-art encryption schemes are used to develop formal security guarantees for new and generalised data fusion algorithms. Industry-standard Kalman filter derivates are modified and existing schemes abstracted such that novel cryptographic notions capturing the required communications can be formalised, while simulations provide an analysis of practicality. Due to the generality of the presented solutions, broad applications are supported, including autonomous vehicle communications, smart sensor networks and distributed localisation.

# Kurzfassung

German abs go here.

# Notation

Complete the notation here.

# 1. Introduction

Sensor data processing, state estimation and data fusion have long been active areas of research and continue to find applications in modern systems [1, 2]. As distributed networks have become more prevalent over the years, greater stress has been put on the need for broadly applicable algorithms that support varying types of measurements, estimate accuracies and communication availabilities [3, 4], finding uses in localisation, weather forecasting, mapping, cooperative computing and cloud computing []. The use of Bayesian estimation methods such as the popular Kalman filter and its non-linear derivatives have become especially prevalent in application due to their recursive, often optimal, estimation properties and their suitability for modelling cross-correlations between local estimates [5, 6]. The handling of these cross-correlations, especially when they are not known in advance, is a common difficulty in state estimation and is tied to the challenges within the field [7]. The methods presented in this thesis and much of the related work in data fusion tasks similarly require the consideration of these challenges.

While the challenges relating to correlation errors are a well-established field of research, widespread advancements in distributed computing and uses of public networks for sensor communication have put a focus on the additional requirements of data privacy and state secrecy in recent years as well [8, 9]. Problems that require both the handling of correlation errors and guaranteeing a level of security for the participants involved are therefore a relevant topic in state estimation today and at the core of the work presented in this thesis. Achieving cryptographic secrecy typically involves hiding transferred information from unauthorised parties and can often be achieved irrespective of the estimation algorithms used by using common symmetric and public-key encryption schemes such as the Advanced Encryption Standard (AES) [10] and the Rivest-Shamir-Adleman cryptosystem (RSA) [11], respectively. These scenarios, however, imply a trust between encrypting and decrypting parties, which cannot always be assumed in distributed environments. In addition, partial computations on encrypted data or the intended leakage of some results are sometimes required for computing final results. This has led to several operation-providing and leakage-supporting encryption schemes [12, 13, 14, 15, 16] suitable for these distributed environments or when fine control over leakage is required. While these cryptographic schemes and notations are applicable in estimation and fusion tasks, the nature of cryptographic analysis in distributed environments, heavily dependent on communication protocols, has meant that developed solutions are often very context-specific, leading to numerous solutions for various estimation scenarios, use-cases and security requirements. This leads us to the current state-of-the-art literature on security-oriented state estimation and data fusion, observable gaps in this literature and the research questions we aim to answer.

## 1.1. State-of-the-Art and Research Questions

To summarise relevant work in security-oriented state estimation and data fusion we first discuss the security and types of data explored in this thesis. Regarding algorithms, we primarily consider stochastic models and outputs. As discussed in section [], Bayesian estimation methods such as the Kalman filter are prevalent due to their applicability and suitability for modelling phenomena accurately. In terms of security, we restrict ourselves to the data confidentiality component of the Confidentiality-Integrity-Availability (CIA) triad []. That is, the primary concern of security in this thesis is that concrete data deemed to be private to some participants remain so and its leakage is formally quantifiable. Data privacy, a related concept, is concerned with stopping the identification of individuals from available information. Although data privacy encompasses data confidentiality, it is a broader topic including communication traffic analyses and external cross-referencing to identify individuals and is not considered in its entirety in this thesis. That said, the terms *data privacy* and *privacy-preserving* are often used in the state-of-the-art to refer to data confidentiality alone and the ability to identify individuals from concrete data available []. The terms will be similarly used throughout this thesis.

Since knowing exact communications between participating parties is required for meaningful cryptographic analysis of transferred data, many existing general estimation algorithms have been restricted in some way to make communication and security easier to discuss. For example, [aristov] presents a distributed Kalman filter, namely an Information filter, where sensor measurements and measurement errors are known only to the measuring sensors while final estimates are leaked to an estimator. The restriction for this to be achieved requires sensors to form a hierarchical communication structure and measurement models to be linear, limiting the otherwise broadly applicable non-linear models or arbitrary communication structures. Another work, [proloc], presents localisation using range-only measurements where measurements and sensor locations, both required for localisation, are kept private to sensors and a centralised estimator while final estimates are available to an external trusted party. Here, no communication structure is enforced but produced estimates provide no error statistics and do not consider a dynamic process model.

In [pwsac, pwsah],

aggregation papers

differential privacy and differentially private Kalman filtering

privacy-preserving optimisation with security based on statistical estimation

added noise estimation

–

privacy-preserving image-based localisation

eavesdropper paper with a secure return channel and a lossier channel for eavesdroppers

GPS

chaotic system paper

physical layer noise paper (similar to chaotic noise paper)

–

The two different approaches, restricting existing broad estimation methods in some ways to make cryptographic analysis plausible and ignoring formal security when assumptions and conclusions are intuitive demonstrate a gap in the existing literature and bring us to the target research topics this thesis aims to explore.

- dot point topics

These broad topics aim to fulfil the goal of generalisable but cryptographically provable estimation and fusion methods in distributed environments and lead to the concrete problems tackled in this work

## 1.2. Contributions

The contributions tackle the research topics in section .. by considering three concrete problems that coincide with the broader problems in the field

## 1.3. Thesis Structure

Each chapter includes relevant related literature to the specific problem and a formal problem formalisation before presenting the novel solutions.

# 2. Preliminaries

When introducing the novel estimation and cryptographic methods in this thesis, we make use of several existing algorithms. In this section, an overview of these methods is given

## 2.1. Estimation Preliminaries

As introduced in section .., the estimation and fusion algorithms considered are Bayesian and based on the Kalman filter and derivatives. The linear KF and linearising EKF are used explicitly and are the focus of the presented preliminaries.

### 2.1.1. Kalman Filter

### 2.1.2. Kalman Filter Optimality

### 2.1.3. Extended Kalman Filter

### 2.1.4. Information Filter

### 2.1.5. Extended Information Filter

### 2.1.6. Fast Covariance Intersection

Covariance Intersection (CI), introduced in [julierNondivergentEstimationAlgorithm1997], provides a consistent state estimate fusion algorithm when cross-correlations are not known. The resulting fused estimate $\underline{\hat{x}}$ and covariance $\mathbf{P}$ can be easily derived from its equations

$$\mathbf{P}^{-1} = \sum_{i=1}^{n} \omega_i \mathbf{P}_i^{-1},\ \mathbf{P}^{-1}\underline{\hat{x}} = \sum_{i=1}^{n} \omega_i \mathbf{P}_i^{-1}\underline{\hat{x}}_i\ . \tag{2.1}$$

Note that (2.1) computes the fusion of the information vectors and information matrices defined in [niehsenInformationFusionBased2002] and reduces the fusion to a weighted sum. Values for weights $\omega_i$ must satisfy

$$\omega_1 + \omega_2 + \cdots + \omega_n = 1,\ 0 \leq \omega_i \leq 1\ , \tag{2.2}$$

which guarantees consistency of the fused estimates. They are chosen in a way to speed up convergence and minimize the error by minimizing a certain specified property of the resulting fused estimate covariance. One such property, the covariance trace, requires

the solution to

$$\underset{\omega_1,\ldots,\omega_n}{\arg\min}\{\operatorname{tr}(\mathbf{P})\} = \underset{\omega_1,\ldots,\omega_n}{\arg\min}\left\{\operatorname{tr}\left(\left(\sum_{i=1}^{n}\omega_i\mathbf{P}_i^{-1}\right)^{-1}\right)\right\} \tag{2.3}$$

for computing weights $\omega_i$. However, minimizing this non-linear cost function can be very computationally costly and has led to the development of faster approximation techniques.

The Fast Covariance Intersection (FCI) algorithm from [niehsenInformationFusion-Based2002] is a non-iterative method for approximating the solution to (2.3) without the loss of guaranteed consistency. It is computed by defining a new constraint

$$\omega_i \operatorname{tr}(\mathbf{P}_i) - \omega_j \operatorname{tr}(\mathbf{P}_j) = 0,\ i,j = 1,2,\ldots,n \tag{2.4}$$

on $\omega_i$ and solving the resulting equations instead. In the two sensor case, this results in the solving of

$$\omega_1 \operatorname{tr}(\mathbf{P}_1) - \omega_2 \operatorname{tr}(\mathbf{P}_2) = 0,\ \omega_1 + \omega_2 = 1\ . \tag{2.5}$$

When computed for $n$ sensors, the highly redundant (2.4) can have its largest linearly independent subset represented by

$$\omega_i \operatorname{tr}(\mathbf{P}_i) - \omega_{i+1} \operatorname{tr}(\mathbf{P}_{i+1}) = 0,\ i = 1,2,\ldots,n-1\ , \tag{2.6}$$

and requires the solution to the linear problem

$$\begin{bmatrix} \mathcal{P}_1 & -\mathcal{P}_2 & 0 & \cdots & 0 \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ 0 & \cdots & 0 & \mathcal{P}_{n-1} & -\mathcal{P}_n \\ 1 & \cdots & 1 & 1 & 1 \end{bmatrix} \begin{bmatrix} \omega_1 \\ \vdots \\ \omega_{n-1} \\ \omega_n \end{bmatrix} = \begin{bmatrix} 0 \\ \vdots \\ 0 \\ 1 \end{bmatrix}\ , \tag{2.7}$$

where we let $\mathcal{P}_i = \operatorname{tr}(\mathbf{P}_i)$.

Our proposed filter aims to solve FCI fusion, namely (2.1) and (2.7), using only encrypted values from each sensor $i$, and leaking only the weight values $\omega_1,\ldots,\omega_n$.

## 2.2. Encryption Preliminaries

Used cryptographic notions and schemes are summarised here. In addition, the encoding of floating point numbers to integers suitable for encryption by the presented schemes is introduced as well.

### 2.2.1. Meeting Cryptographic Notions

### 2.2.2. Paillier Homomorphic Encryption Scheme

The Paillier encryption scheme [paillierPublicKeyCryptosystemsBased1999] is an additively homomorphic encryption scheme that bases its security on the decisional composite residuosity assumption (DCRA) and meets the security notion of IND-CPA. Key generation of the Paillier scheme is performed by choosing two sufficiently large primes $p$ and $q$, and computing $N = pq$. A generator $g$ is also required for encryption, which is often set to $g = N + 1$ when $p$ and $q$ are of equal bit length [katzIntroductionModernCryptography2008]. The public key is defined by $(N, g)$ and the secret key by $(p, q)$.

Encryption of a plaintext message $a \in \mathbb{Z}_N$, producing ciphertext $c \in \mathbb{Z}_{N^2}^*$, is computed by

$$c = g^a \rho^N \pmod{N^2} \tag{2.8}$$

for a randomly chosen $\rho \in \mathbb{Z}_N$. Here, $\rho^N$ can be considered the noise term which hides the value $g^a \pmod{N^2}$, which due to the scheme construction, is an easily computable discrete logarithm. The decryption of a ciphertext is computed by

$$a = \frac{L(c^\lambda \pmod{N^2})}{L(g^\lambda \pmod{N^2})} \pmod{N} \tag{2.9}$$

where $\lambda = \mathsf{lcm}(p-1, q-1)$ and $L(\psi) = \frac{\psi - 1}{N}$.

In addition to encryption and decryption, the following homomorphic functions are provided by the Paillier scheme. $\forall a_1, a_2 \in \mathbb{Z}_N$,

$$\mathcal{D}(\mathcal{E}(a_1)\mathcal{E}(a_2) \pmod{N^2}) = a_1 + a_2 \pmod{N}, \tag{2.10}$$

$$\mathcal{D}(\mathcal{E}(a_1) g^{a_2} \pmod{N^2}) = a_1 + a_2 \pmod{N}, \tag{2.11}$$

$$\mathcal{D}(\mathcal{E}(a_1)^{a_2} \pmod{N^2}) = a_1 a_2 \pmod{N}. \tag{2.12}$$

### 2.2.3. Joye-Libert Aggregation Scheme

The Joye-Libert privacy-preserving aggregation scheme [joyeScalableSchemePrivacyPreserving2013] is a scheme defined on time-series data and meets the security notion of Aggregator Obliviousness (AO) [shiPrivacyPreservingAggregationTimeSeries2011]. Similarly to the Paillier scheme, it bases its security on the DCRA. A notable difference to a public-key encryption scheme is its need for a trusted party to perform the initial key generation and distribution.

Key generation is computed by first choosing two equal-length and sufficiently large primes $p$ and $q$, and computing $N = pq$. A hash function $H : \mathbb{Z} \to \mathbb{Z}_{N^2}^*$ is defined and the public key is set to $(N, H)$. $n$ private keys are generated by choosing $sk_i$, $i \in \{1, \ldots, n\}$, uniformly from $\mathbb{Z}_{N^2}$ and distributing them to $n$ participants (whose values are to be

aggregated), while the last key is set as

$$sk_0 = -\sum_{i=1}^{n} sk_i\,, \tag{2.13}$$

and sent to the aggregator.

Encryption of plaintext $a_i^{(t)} \in \mathbb{Z}_N$ to ciphertext $c_i^{(t)} \in \mathbb{Z}_{N^2}$ at instance $t$ is computed by user $i$ as

$$c_i^{(t)} = (N+1)^{a_i^{(t)}} H(t)^{sk_i} \pmod{N^2}\,. \tag{2.14}$$

Here, we can consider $H(t)^{sk_i}$ the noise term which hides the easily computable discrete logarithm $g^{a_i^{(t)}} \pmod{N^2}$, where $g = N+1$ (as with the Paillier scheme above).

When all encryptions $c_i^{(t)}$, $i \in \{1, \ldots, n\}$ are sent to the aggregator, summation and decryption of the aggregated sum are computed by the functions

$$c^{(t)} = H(t)^{sk_0} \prod_{i=1}^{n} c_i^{(t)} \pmod{N^2} \tag{2.15}$$

and

$$\sum_{i=1}^{n} a_i^{(t)} = \frac{c^{(t)} - 1}{N} \pmod{N}\,. \tag{2.16}$$

Correctness follows from $\sum_{i=0}^{n} sk_i = 0$, and thus

$$\begin{aligned}
&H(t)^{sk_0} \prod_{i=1}^{n} c_i^{(t)} \pmod{N^2} \\
\equiv &H(t)^{sk_0} \prod_{i=1}^{n} (N+1)^{a_i^{(t)}} H(t)^{sk_i} \pmod{N^2} \\
\equiv &H(t)^{\sum_{j=0}^{n} sk_j} \prod_{i=1}^{n} (N+1)^{a_i^{(t)}} \pmod{N^2} \\
\equiv &(N+1)^{\sum_{i=1}^{n} a_i^{(t)}} \pmod{N^2}\,,
\end{aligned}$$

removing all noise terms.

### 2.2.4. Lewi Order-Revealing Encryption Scheme

For ORE, we use the Lewi symmetric-key Left-Right ORE scheme as it has the added property of only allowing certain comparisons between cyphertexts. This property can be used to decide which values may not be compared, which will be shown in the section ... It is described as follows: two encryption functions allow integers to be encrypted as

either a "Left" ($L$) or "Right" ($R$) encryption by

$$\begin{aligned} \mathrm{enc}^{\mathrm{L}}_{\mathrm{ORE}}(\mathrm{k},\mathrm{x}) &= \mathcal{E}^{\mathrm{L}}_{\mathrm{ORE},\mathrm{k}}(\mathrm{x}) \ , \\ \mathrm{enc}^{\mathrm{R}}_{\mathrm{ORE}}(\mathrm{k},\mathrm{y}) &= \mathcal{E}^{\mathrm{R}}_{\mathrm{ORE},\mathrm{k}}(\mathrm{y}) \ , \end{aligned} \tag{2.17}$$

and only comparisons between an $L$ and an $R$ encryption are possible, by

$$\mathrm{cmp}_{\mathrm{ORE}}(\mathcal{E}^{\mathrm{L}}_{\mathrm{ORE}}(\mathrm{x}),\ \mathcal{E}^{\mathrm{R}}_{\mathrm{ORE}}(\mathrm{y})) = \mathrm{cmp}(\mathrm{x},\mathrm{y}) \ . \tag{2.18}$$

Note that no decryption function is provided as only encryptions are required to provide a secure comparison. The Lewi ORE encryption scheme provides security against the simulation-based security model [chenettePracticalOrderRevealingEncryption2016] but is not secure against the IND-OCPA model.

### 2.2.5. Encoding Numbers for Encryption

In both the Paillier and Joye-Libert schemes, as well as the one we introduce, meaningful inputs $a$ are bounded to $a \in \mathbb{Z}_N$. For this reason, real-valued estimation variables require quantisation and integer mapping for encryption and aggregation. We will rely on a generalised Q number encoding [oberstarFixedPointRepresentationFractional2007] due to implementation simplicity and applicability.

We will consider a subset of rational numbers in terms of a range $M \in \mathbb{N}$ and fractional precision $\phi \in \mathbb{N}$. This contrasts with the common definition in terms of total and fractional bits [oberstarFixedPointRepresentationFractional2007, schulzedarupEncryptedCooperativeControl2019, farokhiSecurePrivateControl2017], but allows for a direct mapping to integer ranges which are not a power of two. A rational subset $\mathbb{Q}_{M,\phi}$ is then given by

$$\mathbb{Q}_{M,\phi} = \left\{ o \,\middle|\, \phi o \in \mathbb{N} \wedge -\left\lfloor \frac{M}{2} \right\rfloor \leq \phi o < \left\lfloor \frac{M}{2} \right\rfloor \right\} , \tag{2.19}$$

and we can quantize any real number $a$ by taking the nearest rational $o \in \mathbb{Q}_{M,\phi}$, that is, $\arg\min_{o \in \mathbb{Q}_{M,\phi}} |a - o|$. In this form, mapping rationals $\mathbb{Q}_{M,\phi}$ to an encryption range $\mathbb{Z}_N$ is achieved by choosing $M = N$ and handling negatives by modulo arithmetic. Additionally, we note that the Q number format requires a precision factor $\phi$ to be removed after each encoded multiplication. This is captured by a third parameter $d$; the number of additional precision factors present in encodings.

The function for *combined* quantisation and encoding, $\mathsf{E}_{M,\phi,d}(a)$, of a given number $a \in \mathbb{R}$ and with an integer range $\mathbb{Z}_M$, precision $\phi$ and scaling for $d$ prior encoded multiplications is given by

$$\mathsf{E}_{M,\phi,d}(a) = \left\lfloor \phi^{d+1} a \right\rceil \pmod{M} . \tag{2.20}$$

Decoding of an integer $u \in \mathbb{Z}_M$, is given by

$$\mathsf{E}^{-1}_{M,\phi,d}(u) = \begin{cases} \dfrac{u \pmod M}{\phi^{d+1}}, & u \pmod M \leq \left\lfloor \dfrac{M}{2} \right\rfloor \\ -\dfrac{M - u \pmod M}{\phi^{d+1}}, & \text{otherwise} \end{cases}. \tag{2.21}$$

This encoding scheme provides the following homomorphic operations,

$$\mathsf{E}_{M,\phi,d}(a_1) + \mathsf{E}_{M,\phi,d}(a_2) \pmod M = \mathsf{E}_{M,\phi,d}(a_1 + a_2) \tag{2.22}$$

and

$$\mathsf{E}_{M,\phi,d}(a_1)\mathsf{E}_{M,\phi,d}(a_2) \pmod M = \mathsf{E}_{M,\phi,d+1}(a_1 a_2)\,, \tag{2.23}$$

noting that when $M = N$, the operations and modulus correspond with those in the Paillier homomorphic operations (2.10), (2.11) and (2.12), and the Joye-Libert sum (2.16).

In general, the choice of a large precision parameter $\phi$ may reduce quantisation errors introduced in (2.20), but risks overflow after too many multiplications. Given the largest number of encoded multiplications, $d_{max}$, and the largest value to be encoded $a_{max}$, the parameter should be chosen such that

$$\left|\phi^{d_{max}+1} a_{max}\right| < \left\lfloor \frac{M}{2} \right\rfloor . \tag{2.24}$$

In practice, $N$ is typically very large ($N > 2^{1024}$) and this condition can be ignored when $M = N$, as $\phi$ can be made sufficiently large to make quantisation errors negligible.

# 3. Estimate Fusion on an Untrusted Cloud

## 3.1. Problem Formulation

In this work, we consider an arbitrary time-varying process defined by its state $\underline{x}_k \in \mathbb{R}^n$ for all timesteps $k \in \mathbb{N}$. This process is estimated by $m$ individual estimators $i$, $1 \leq i \leq m$, each producing a state estimate and an estimate error covariance,

$$\underline{\hat{x}}_{k,i} \in \mathbb{R}^n \text{ and } \mathbf{P}_{k,i} \in \mathbb{R}^{n \times n}\,, \tag{3.1}$$

respectively, at every timestep $k$. Our aim at every timestep is for all estimates $\underline{\hat{x}}_{k,i}$ and errors $\mathbf{P}_{k,i}$, $1 \leq i \leq m$, to be sent to a cloud server where a fused estimate and error covariance,

$$\underline{\hat{x}}_{k,\mathsf{fus}} \in \mathbb{R}^n \text{ and } \mathbf{P}_{k,\mathsf{fus}} \in \mathbb{R}^{n \times n}\,, \tag{3.2}$$

respectively, are computed and are consistent with the estimates in (3.1).

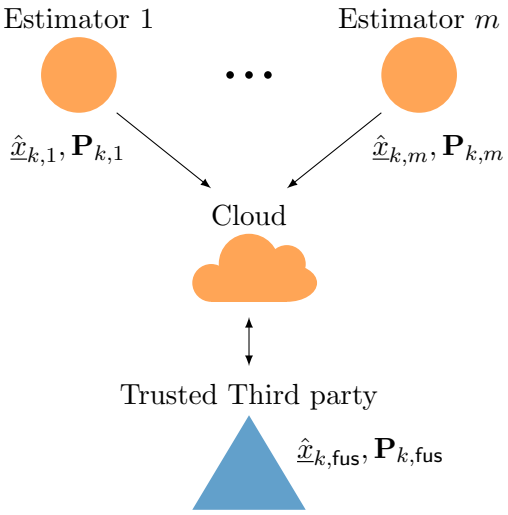
Simultaneously, we consider the desired security of the fusion process. The fusing cloud and estimators are treated as *honest-but-curious*, that is, they follow protocols correctly but may use learned information for malicious gain, and a trusted third party exists that can query the cloud at any timestep $k$ to obtain the fusion results in (3.2). We aim for the cloud, estimators and eavesdroppers to learn no additional information from observed estimates and fusions, (3.1) and (3.2), respectively, beyond their local estimates. To guarantee this, cryptographic *Indistinguishability under the Chosen Plaintext Attack* (IND-CPA) [katzIntroductionModernCryptography2008] is desired for all transmitted and processed information. This is in line with common security goals in the field and is suitable for homomorphic encryption. The communications of this scenario are summarised graphically in figure **??**.

## 3.2. Related Literature

## 3.3. Confidential Cloud Fusion Leaking Fusion Weights

### 3.3.1. Two-sensor Secure Fast Covariance Intersection

In this section, we will introduce the Secure FCI (SecFCI) fusion algorithm for the two-sensor case, before extending it to the $n$ sensor case in section [sec:multi_secfci]. The network model we consider is described in section [subsec:problem_formulation], where sensors are capable of running local estimators, as well as the PHE and ORE encryption schemes from section [sec:encryption]. Each sensor $i$ computes its state estimate $\underline{\hat{x}}_i$ and covariance matrix $\mathbf{P}_i$ and sends relevant encrypted information to an untrusted cloud




fusion center. The querying party is the key holding party and generates the PHE public key $pk$, secret key $sk$, and ORE symmetric key $k$. $pk$ is made available to all parties in the network, and $k$ is made available to the sensors only, via any standard public-key scheme such as RSA [rivestMethodObtainingDigital1978]. When encrypting with ORE key $k$, individual sensors are limited to using only $L$ or $R$ ORE encryption to reduce local information leakage. Thus, consecutive ORE encryptions from any sensor cannot be used to infer local information directly, and can only be compared to encryptions from sensors using the alternate ORE encryption.

From [eqn:ci_cov_estimate], we can see that both CI fusion equations can be computed on PHE encryptions of sensor information vectors and information matrices, given valid unencrypted values for each $\omega_i$. For this reason, we allow the leakage of all weights $\omega_i$. Thus, in the two-sensor case, homomorphic fusion is computed by

$$\mathcal{E}(\mathbf{P}^{-1}) = \mathcal{E}(\mathbf{P}_1^{-1})^{\omega_1}\mathcal{E}(\mathbf{P}_2^{-1})^{(1-\omega_1)} \tag{3.3}$$

and

$$\mathcal{E}(\mathbf{P}^{-1}\hat{\underline{x}}) = \mathcal{E}(\mathbf{P}_1^{-1}\hat{\underline{x}}_1)^{\omega_1}\mathcal{E}(\mathbf{P}_2^{-1}\hat{\underline{x}}_2)^{(1-\omega_1)} \quad , \tag{3.4}$$

where we note that $\omega_2 = 1-\omega_1$ due to the CI requirement [eqn:ci_omega_sum_bound]. We also note that in [eqn:paillier_ci_cov] and [eqn:paillier_ci_estimate], each resulting value will have exactly one encoding multiplication factor to remove, and can be decoded exactly by using [eqn:qmn_mult_decode].

All that remains for computing CI homomorphically, in the two-sensor case, is the calculation of parameter $\omega_1$. For this, we approximate the solution to FCI. Since our encoding scheme in section [subsec:encoding] does not allow division, the exact result of [eqn:fci_2sen_omega_sum_eq] is approximated. This is accomplished by discretizing $\omega_i$ by step-size $s$, such that $s < 1$ and $p = 1/s \in \mathbb{Z}$, and approximating [eqn:fci_2sen_omega_sum_eq] with ORE. An ordered discretization of values $\omega^{(x)}$ is de-

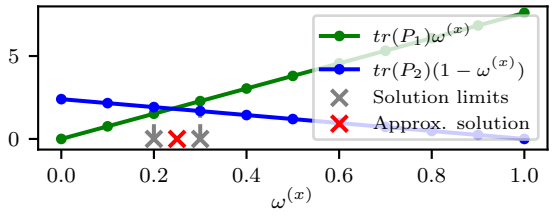



Figure 3.1.: Approximation of $\omega_1$ with discretization step-size $s = 0.1$. Only comparisons between line points are used.

fined by

$$[\omega^{(1)}, \ldots, \omega^{(p)}] = [0, s, \ldots, 1-s, 1] \ , \tag{3.5}$$

and computed by each sensor $i$. Each $\omega^{(x)}$ is multiplied by $\mathrm{tr}(\mathbf{P}_i)$ and encrypted with ORE key $k$. Sensor 1's list is defined by

$$[\mathcal{E}^L_{ORE}(\omega^{(1)}\,\mathrm{tr}(\mathbf{P}_1)), \ldots, \mathcal{E}^L_{ORE}(\omega^{(p)}\,\mathrm{tr}(\mathbf{P}_1))] \ , \tag{3.6}$$

and similarly sensor 2's by

$$[\mathcal{E}^R_{ORE}(\omega^{(1)}\,\mathrm{tr}(\mathbf{P}_2)), \ldots, \mathcal{E}^R_{ORE}(\omega^{(p)}\,\mathrm{tr}(\mathbf{P}_2))] \ . \tag{3.7}$$

Note that Sensor 1 uses only $L$ ORE while sensor 2 uses only $R$ ORE, and that both lists are ordered. Lists (3.6) and (3.7) are sent alongside PHE encryptions of local information vector and information matrix estimates to the fusion center which uses them to estimate the FCI values of $\omega_1$ and $\omega_2$.

From [eqn:fci_2sen_omega_sum_eq] we know that $\omega_1$ must satisfy

$$\omega_1\,\mathrm{tr}(\mathbf{P}_1) = (1-\omega_1)\,\mathrm{tr}(\mathbf{P}_2) \ . \tag{3.8}$$

If we reverse (3.7), we obtain a list equivalent to one with values $\mathcal{E}^R_{ORE}((1-\omega^{(x)})\,\mathrm{tr}(\mathbf{P}_2))$ for each discretization step $x$. When the reversed list is decrypted and plotted over (3.6) the intersection gives the solution to (3.8) and therefore, (2.5). However, (3.6) and reversed (3.7) consist of $L$ and $R$ ORE encryptions respectively, and the intersection must be approximated by locating consecutive $\omega^{(x)}$ discretizations where the sign of comparisons changes. This can be seen in Fig. 3.1, and can be performed in $O(\log p)$ ORE comparisons using a binary search. Consecutive $\omega^{(x)}$ and $\omega^{(x+1)}$ for which list comparisons differ can be used to estimate the true intersection, and $\omega_1$, by

$$\omega_1 \approx 0.5(\omega^{(x)} + \omega^{(x+1)}) \ . \tag{3.9}$$

In the case a comparison returns equality, the exact value of $\omega^{(x)}$ can be taken to be $\omega_1$.

The fusion center can then use its values for $\omega_1$ and $\omega_2 = 1-\omega_1$ and the received PHE

encryptions of local information vectors and information matrices to compute (3.3) and (3.4).

### 3.3.2. Multi-sensor Secure Fast Covariance Intersection

When computing the SecFCI fusion for $n$ sensors, we solve (2.1) homomorphically by computing

$$\mathcal{E}(\mathbf{P}^{-1}) = \mathcal{E}(\mathbf{P}_1^{-1})^{\omega_1} \cdots \mathcal{E}(\mathbf{P}_n^{-1})^{\omega_n} \tag{3.10}$$

and

$$\mathcal{E}(\mathbf{P}^{-1}\underline{\hat{x}}) = \mathcal{E}(\mathbf{P}_1^{-1}\underline{\hat{x}}_1)^{\omega_1} \cdots \mathcal{E}(\mathbf{P}_n^{-1}\underline{\hat{x}}_n)^{\omega_n} \ . \tag{3.11}$$

As with the two-sensor case, encoded results from (3.10) and (3.11) contain exactly one multiplication factor to remove and can be decoded exactly with [eqn:qmn_mult_decode]. Again we are just left with the task of computing the plaintext weights $\omega_1, \ldots, \omega_n$.

Our approach to the $n$ sensor case is to solve all $n-1$ conditions in (2.6) using the two-sensor method, and combining partial solutions to compute the final result. When we consider a Euclidean dimension for each $\omega_i$, partial solutions can be considered geometrically as hyperplanes of $n-2$ dimension, over the $n-1$ dimensional solution space given by (2.2).

This can be visualized in the three sensor case, which requires solving partial solutions

$$\omega_1 \operatorname{tr}(\mathbf{P}_1) - \omega_2 \operatorname{tr}(\mathbf{P}_2) = 0, \ \omega_1 + \omega_2 = 1 - \omega_3 \tag{3.12}$$

and

$$\omega_2 \operatorname{tr}(\mathbf{P}_2) - \omega_3 \operatorname{tr}(\mathbf{P}_3) = 0, \ \omega_2 + \omega_3 = 1 - \omega_1 \ . \tag{3.13}$$

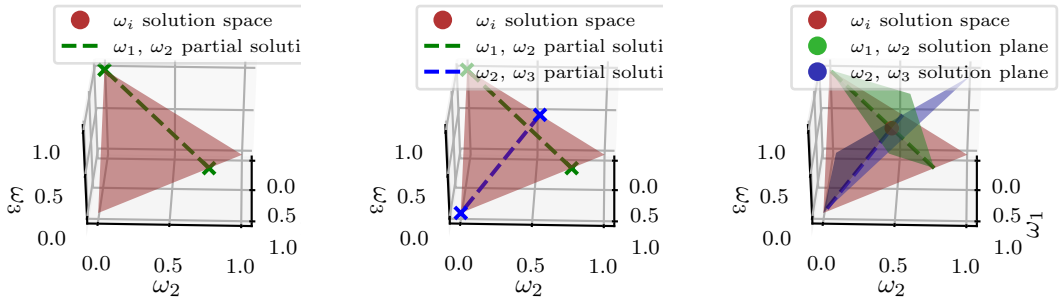
We can use the two-sensor method from section 3.3.1 to solve (3.12) exactly when $\omega_3 = 0$, and know that when $\omega_3 = 1$, then $\omega_1 = \omega_2 = 0$. These two points are enough to define the two-dimensional partial solution (3.12) which can be seen plotted over the possible solution space in Fig. 3.2(a). Fig. 3.2(b) shows both partial solutions (3.12) and (3.13) plotted over the solution space. The final solution from all partial solutions is computed by finding their intersection. This can be seen in Fig. 3.2(b) as the intersection of the $(\omega_1, \omega_2)$ and $(\omega_2, \omega_3)$ partial solution lines.

To simplify computing the partial solution intersection, we define equivalent planes for each of the partial solutions, perpendicular to the solution space, in the form

$$a_1\omega_1 + a_2\omega_2 + a_3\omega_3 + a_4 = 0 \ , \tag{3.14}$$

and solve the resulting linear system for finding the intersection of all planes and the solution space. This is given by

$$\begin{bmatrix} a_1^{(1)} & a_2^{(1)} & a_3^{(1)} \\ a_1^{(2)} & a_2^{(2)} & a_3^{(2)} \\ 1 & 1 & 1 \end{bmatrix} \begin{bmatrix} \omega_1 \\ \omega_2 \\ \omega_3 \end{bmatrix} = \begin{bmatrix} a_3^{(1)} \\ a_4^{(2)} \\ 1 \end{bmatrix} \ , \tag{3.15}$$




(a) Partial solution to (3.12). (b) Partial solutions to (3.12) and (3.13). (c) Partial solutions as planes.

Figure 3.2.: Partial solutions over $\omega_1$, $\omega_2$, and $\omega_3$ solution space.

where $a_i^{(j)}$ denotes parameter $i$ of partial solution $j$, and has been shown visually in Fig. 3.2(c).

In the $n$ sensor case, we can similarly solve partial solutions by first using the method from section 3.3.1 to solve equations with two parameters $\omega_k$ and $\omega_{k+1}$ when letting all $\omega_i = 0,\ i \neq k, k+1$. For each equation we can then compute remaining partial solution points at $\omega_i = 1,\ i \neq k, k+1$ with $\omega_j = 0,\ j \neq i$. Perpendicular hyperplanes can then be similarly defined in the form

$$a_1\omega_1 + \cdots + a_n\omega_n + a_{n+1} = 0 \ . \tag{3.16}$$

Due to their inherent orthogonality, and that all meaningful covariance traces are strictly positive, the $n-1$ partial solution hyperplanes are guaranteed to intersect at exactly one point. The hyperplane intersection results in the linear system

$$\begin{bmatrix} a_1^{(1)} & a_2^{(1)} & \cdots & a_n^{(1)} \\ \vdots & \vdots & \ddots & \vdots \\ a_1^{(n-1)} & a_2^{(n-1)} & \cdots & a_n^{(n-1)} \\ 1 & 1 & \cdots & 1 \end{bmatrix} \begin{bmatrix} \omega_1 \\ \vdots \\ \omega_{n-1} \\ \omega_n \end{bmatrix} = \begin{bmatrix} a_{n+1}^{(1)} \\ \vdots \\ a_{n+1}^{(n-1)} \\ 1 \end{bmatrix} \ , \tag{3.17}$$

and gives the solution to the SecFCI $\omega_i$ weights.

As all $O(n \log p)$ ORE comparisons are done between sequential sensors $i$ and $i+1$, $L$ and $R$ ORE encryptions can be used to the same effect as for the two-sensor case. The ORE ordered list sent from each sensor $i$ is given by

$$\begin{aligned} &[\mathcal{E}^L_{ORE}(\omega^{(1)} \operatorname{tr}(\mathbf{P}_i)), \ldots, \mathcal{E}^L_{ORE}(\omega^{(p)} \operatorname{tr}(\mathbf{P}_i))],\ i \text{ odd} \\ &[\mathcal{E}^R_{ORE}(\omega^{(1)} \operatorname{tr}(\mathbf{P}_i)), \ldots, \mathcal{E}^R_{ORE}(\omega^{(p)} \operatorname{tr}(\mathbf{P}_i))],\ i \text{ even}. \end{aligned} \tag{3.18}$$

When combining (3.18) with PHE encryptions of local information vectors and informa-

Table 3.1.: Computation complexity of encryption operations.

| Operation | Complexity |
|---|---|
| Paillier enc. | $O(\log^3 N)$ |
| Paillier dec. | $O(\log^3 N)$ |
| Paillier add. | $O(\log^2 N)$ |
| Paillier scalar mult. | $O(\log^3 N)$ |
| Lewi $L$ enc. | $O(\log^2 N)$ |
| Lewi $R$ enc. | $O(\log^2 N)$ |
| Lewi comp. | $O(\log^2 N)$ |

tion matrices, SecFCI can be computed entirely homomorphically by (3.10) and (3.11).

Briefly considering the security of our scheme, we note that any leaked information from ORE lists (3.18), as described in [chenettePracticalOrderRevealingEncryption2016], can be considered a subset of knowing the estimated fusion weights $\omega_1, \ldots, \omega_n$, which specify relative sizes of sensor covariance traces, and we already consider public. Thus only IND-CPA and IND-OCPA (after accounting for leakage through public weights) encryptions are made available to the fusion centre.

### 3.3.3. Computational Complexity

Given the state estimates and estimate errors at each sensor, we wish to show the computational complexity of the SecFCI algorithm for the $n$ sensor case. We will assume that both Lewi ORE and Paillier PHE schemes use the same length security parameter (and equivalently key size), such that $\lambda_{Lewi} = \lambda_{Paillier} = \log N$, where $\lambda_s$ represents encryption scheme $s$'s security parameter, and $N$ the Paillier modulus and encryptable integer limit. We also note the distinction between floating-point or small integer operations, which are typically treated as having $O(1)$ runtime, and large integer operations whose complexities are dependent on bit length. While architectures exist for speeding up encryption operations [gueronIntelAdvancedEncryption2010], we consider software implementations and treat large integer operations in terms of bit operations explicitly.

From [paillierPublicKeyCryptosystemsBased1999,lewiOrderRevealingEncryptionNew2016], and the assumptions made above, we have summarized the operation complexities of the two schemes in Table 3.1. In contrast to some current FHE schemes, these operations are of a much lower complexity than [vandijkFullyHomomorphicEncryption2010a], which has complexity $O(\lambda^{10})$ for integer operations, and [stehleFasterFullyHomomorphic2010], which computes single bit operations in $O(\lambda^{3.5})$ adding significant overhead for integer arithmetic.

Finally, applying the operations from Table 3.1 to the SecFCI algorithm, we summarize the total complexity of SecFCI at the sensors and the fusion centre in Table 3.2, with the unencrypted complexities of FCI shown for reference.

Table 3.2.: Computation complexity at sensors and fusion centre.

| | **FCI** | **SecFCI** |
|---|---|---|
| Sensors | $O(1)$ | $O\left(p\log^2 N + \log^3 N\right)$ |
| Fusion | $O(n^3)$ | $O\left(n\log p\log^2 N + n\log^3 N + n^3\right)$ |

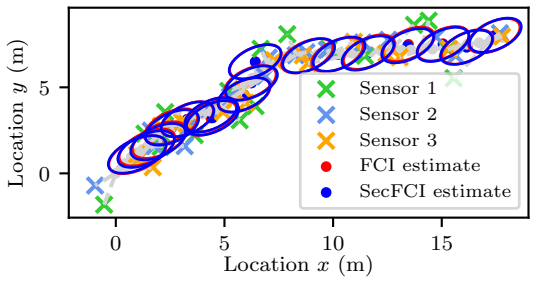



Figure 3.3.: Tracking simulation comparing SecFCI and FCI.

### 3.3.4. Simulation Results

We have implemented a simulation to demonstrate the accuracy of SecFCI approximating FCI. Three sensors independently measure a constant-speed linear process and simultaneously run a Kalman filter on their measurements. Estimates are sent both encrypted and unencrypted to a fusion centre that computes the SecFCI and FCI fusions on the received data respectively. Encrypted estimates are comprised of PHE encryptions of the information vector and information matrix, $\mathcal{E}(\mathbf{P}_i^{-1}\underline{\hat{x}}_i)$ and $\mathcal{E}(\mathbf{P}_i^{-1})$, in addition to the ORE list given by (3.18) with discretization step $s = 0.1$. Unencrypted estimates consist of the state estimate $\underline{\hat{x}}_i$ and covariance $\mathbf{P}_i$. The trajectory and fused estimates are shown in Fig. 3.3.

To derive an upper bound on the accuracy difference between SecFCI and FCI, we note the two factors which introduce inconsistency between the two methods: the encoding method from section 3.3.3, and the difference in fusion weights. Due to the possibility of choosing sufficiently large integer and fractional bit lengths $i$ and $f$, we will only consider the error caused by the difference in weights. We will treat this error as the distance between respective weight vectors

$$\begin{aligned}\underline{\omega}_{SecFCI} &= (\omega_{1,SecFCI},\ldots,\omega_{n,SecFCI})\\ \underline{\omega}_{FCI} &= (\omega_{1,FCI},\ldots,\omega_{n,FCI})\ ,\end{aligned} \tag{3.19}$$

where $\omega_{i,s}$ denotes weight $\omega_i$ from algorithm $s$. From section 3.3.1 we see that the largest difference $|\omega_{i,FCI} - \omega_{i,SecFCI}|$ is strictly bounded by $s/2$. As shown in section 3.3.2, when more sensors are involved, a tighter bound on this difference is dependent on the value of $\underline{\omega}_{i,FCI}$, but will remain strictly bounded by $s/2$. Therefore, we can give a strict upper

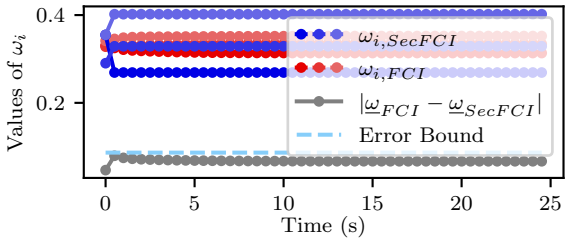



Figure 3.4.: $\underline{\omega}_{SecFCI}$ and $\underline{\omega}_{FCI}$ components.

bound on the distance between weight vectors as

$$|\underline{\omega}_{FCI} - \underline{\omega}_{SecFCI}| < 0.5\sqrt{ns^2} \ . \tag{3.20}$$

Finally, components of $\omega_{i,SecFCI}$, $\omega_{i,FCI}$ and the errors $|\underline{\omega}_{FCI} - \underline{\omega}_{SecFCI}|$, have been plotted over time in Fig. 3.4, and show the computed error bound when $n = 3$ and $s = 0.1$.

## 3.4. Confidential Cloud Fusion Without Leakage

With the problem and preliminaries introduced, we can now present our encrypted FCI method that leaks no estimator information to the fusing cloud. The core idea behind the method is to postpone the evaluation of operations that cannot be performed homomorphically until partial results are queried and decrypted by the key-holding third party. The remaining operations can then be evaluated on unencrypted inputs to produce the correct results.

First, we note that the FCI fusion equations [eqn:ci_cov_fusion] and [eqn:ci_est_fusion] can be rearranged and substituted with weights [eqn:fci_weights] to obtain the equations

$$\mathbf{P}_{k,\mathsf{fus}} = \left( \left( \sum_{i=1}^{m} \frac{1}{\mathrm{tr}(\mathbf{P}_{k,i})} \right)^{-1} \sum_{i=1}^{m} \frac{1}{\mathrm{tr}(\mathbf{P}_{k,i})} \mathbf{P}_{k,i}^{-1} \right)^{-1} \tag{3.21}$$

and

$$\hat{\underline{x}}_{k,\mathsf{fus}} = \mathbf{P}_{k,\mathsf{fus}} \left( \sum_{i=1}^{m} \frac{1}{\mathrm{tr}(\mathbf{P}_{k,i})} \right)^{-1} \sum_{i=1}^{m} \frac{1}{\mathrm{tr}(\mathbf{P}_{k,i})} \mathbf{P}_{k,i}^{-1} \hat{\underline{x}}_{k,i} \, . \tag{3.22}$$

In this form, innermost summations

$$\sum_{i=1}^{m} \frac{1}{\operatorname{tr}(\mathbf{P}_{k,i})}\,,\ \sum_{i=1}^{m} \frac{1}{\operatorname{tr}(\mathbf{P}_{k,i})}\mathbf{P}_{k,i}^{-1} \text{ and } \sum_{i=1}^{m} \frac{1}{\operatorname{tr}(\mathbf{P}_{k,i})}\mathbf{P}_{k,i}^{-1}\hat{\underline{x}}_{k,i} \tag{3.23}$$

combine information from individual estimators $i$ and are computable homomorphically given suitable encryptions. Encryptions of these sums can then be decrypted by the key-holding third party, before remaining inversions and multiplications in (3.21) and (3.22) can be computed to obtain the final results. To depict this process, pseudocode for the encryption at estimators, fusion at the cloud and decryption by the third party are provided in algorithms 1, 2 and 3, respectively.

**Algorithm 1** Estimator Encryption

**procedure** ESTIMATE($i$, $k$, $\mathsf{pk}$, $\phi$)
  Estimate $\hat{\underline{x}}_{k,i}$ locally
  Estimate $\mathbf{P}_{k,i}$ locally
  ▷ Public key is encoding and encryption modulus
  $N \leftarrow \mathsf{pk}$
  ▷ Encode scaling, covariance and estimate components
  $\tilde{s}_{k,i} \leftarrow \mathsf{E}_{N,\phi}\left(\frac{1}{\operatorname{tr}(\mathbf{P}_{k,i})}\right)$
  $\tilde{\mathbf{C}}_{k,i} \leftarrow \mathsf{E}_{N,\phi}\left(\frac{1}{\operatorname{tr}(\mathbf{P}_{k,i})}\mathbf{P}_{k,i}^{-1}\right)$
  $\tilde{\underline{e}}_{k,i} \leftarrow \mathsf{E}_{N,\phi}\left(\frac{1}{\operatorname{tr}(\mathbf{P}_{k,i})}\mathbf{P}_{k,i}^{-1}\hat{\underline{x}}_{k,i}\right)$
  ▷ Encrypt scaling, covariance and estimate components
  $s_{k,i} \leftarrow \mathcal{E}_{\mathsf{pk}}\left(\tilde{s}_{k,i}\right)$
  $\mathbf{C}_{k,i} \leftarrow \mathcal{E}_{\mathsf{pk}}\left(\tilde{\mathbf{C}}_{k,i}\right)$
  $\underline{e}_{k,i} \leftarrow \mathcal{E}_{\mathsf{pk}}\left(\tilde{\underline{e}}_{k,i}\right)$
  Send $s_{k,i}$, $\mathbf{C}_{k,i}$ and $\underline{e}_{k,i}$ to fusing cloud
**end procedure**

*Remark.* Along with allowing the summations to be performed homomorphically on the cloud, we note that this form of the FCI also allows the cloud's partial fusion operations to be evaluated sequentially. This can be seen in algorithm 2, where individual components $s_{k,i}$, $\mathbf{C}_{k,i}$ and $\underline{e}_{k,i}$ from each estimator can continue to be aggregated as additional estimators send their estimate information. This, in turn, supports the dynamic joining and leaving of estimators in the network without affecting the cloud or the operations

**Algorithm 2** Cloud Fusion

**procedure** FUSE($k$, $\mathsf{pk}$)
  Receive $s_{k,i}$, $\mathbf{C}_{k,i}$ and $\underline{e}_{k,i}$ for all $1 \leq i \leq m$
  $N \leftarrow \mathsf{pk}$
  $s_k \leftarrow \prod_{i=1}^{m} s_{k,i} \pmod{N^2}$
  $\mathbf{C}_k \leftarrow \otimes_{i=1}^{m} \mathbf{C}_{k,i} \pmod{N^2}$
  $\underline{e}_k \leftarrow \otimes_{i=1}^{m} \underline{e}_{k,i} \pmod{N^2}$
  Store $s_k$, $\mathbf{C}_k$ and $\underline{e}_k$ in case of query
**end procedure**

**Algorithm 3** Fusion Query

**procedure** GETRESULT($k$, $\mathsf{pk}$, $\mathsf{sk}$, $\phi$)
  Query and receive $s_k$, $\mathbf{C}_k$ and $\underline{e}_k$ from fusing cloud
  $N \leftarrow \mathsf{pk}$
  ▷ Decrypt
  $\tilde{s}_k \leftarrow \mathcal{D}_{\mathsf{sk}}(s_k)$
  $\tilde{\mathbf{C}}_k \leftarrow \mathcal{D}_{\mathsf{sk}}(\mathbf{C}_k)$
  $\underline{\tilde{e}}_k \leftarrow \mathcal{D}_{\mathsf{sk}}(\underline{e}_k)$
  ▷ Decode
  $\bar{s}_k \leftarrow \mathsf{E}_{N,\phi}^{-1}(\tilde{s}_k)$
  $\bar{\mathbf{C}}_k \leftarrow \mathsf{E}_{N,\phi}^{-1}\left(\tilde{\mathbf{C}}_k\right)$
  $\underline{\bar{e}}_k \leftarrow \mathsf{E}_{N,\phi}^{-1}(\underline{\tilde{e}}_k)$
  ▷ Compute Fusion
  $\mathbf{P}_{k,\mathsf{fus}} \leftarrow \left(\bar{s}_k^{-1} \cdot \bar{\mathbf{C}}_k\right)^{-1}$
  $\underline{\hat{x}}_{k,\mathsf{fus}} \leftarrow \mathbf{P}_{k,\mathsf{fus}} \cdot \bar{s}_k^{-1} \cdot \underline{\bar{e}}_k$
  **return** $\underline{\hat{x}}_{k,\mathsf{fus}}$, $\mathbf{P}_{k,\mathsf{fus}}$
**end procedure**

Table 3.3.: Computation complexity of encryption operations.

| Operation | Complexity |
|---|---|
| Encryption | $O(\log^3 N)$ |
| Decryption | $O(\log^3 N)$ |
| Addition | $O(\log^2 N)$ |
| Scalar mult. | $O(\log^3 N)$ |

Table 3.4.: Computation complexity at parties during fusion.

| | FCI | Our Method |
|---|---|---|
| Estimator | $O(1)$ | $O\left(n^2 \log^3 N\right)$ |
| Fusion | $O(mn^3)$ | $O\left(mn^2 \log^2 N\right)$ |
| Third party | $O(1)$ | $O\left(n^2 \log^3 N + n^3\right)$ |

of a trusted third party. The security implications of such an extension are discussed further in section 3.5.1.

## 3.5. Complexity

The method described provides a level of security when relying on an untrusted cloud for fusing estimates. However, the added reliance on an encryption scheme and the additional computations at the third party intuitively increase the computational complexity of the algorithm and the required capabilities of participating parties. Here, we present the complexity of operations during fusion, required by each party at every timestep $k$. We assume encoding and decoding operations have complexity $O(1)$ (due to their comparative insignificance when compared to associated encryption and decryption operations) and use [12] to obtain the encryption complexities of the Paillier encryption scheme in table 3.3, with security parameter $\lambda = \log N$. In table 3.4, we compare the complexities of the unencrypted FCI algorithm and the method presented in this work. It can be seen that the burden of computation is greatly increased at the estimators and the third party, in particular when dimension $n$ is large and when a long encryption key $N$ is used. Naturally, an application of the proposed method would need to consider these requirements in terms of computation time and required hardware.

### 3.5.1. Security Analysis

The provable security of the presented method is relatively straightforward. Our aim for IND-CPA security of all information received by the cloud, sent by the estimators or observable by eavesdroppers is achieved by the homomorphic Paillier encryption scheme. Since all transmitted information is encrypted and the cloud, estimators and eavesdroppers do not hold the secret key $\mathsf{sk}$, IND-CPA is met at all parties.

We note, however, an implicit assumption made when encrypting multidimensional

data element-wise. While individual elements are indistinguishable, element-wise encryption does not encrypt the estimate's dimension $n$, which remains implicitly public. While existing methods allow the complete homomorphic encryption of vectors [alexandruPrivateWeightedSum2020], they are left for future work and considered beyond the scope of this work. Instead, we acknowledge the implicit leakage of $n$ and note that, while this may leak information about the fusion's use case, state estimates remain hidden. Additionally, intuitive extensions to the scheme, such as the dynamic joining and leaving of estimators in remark 3.4, may introduce further implicit leakages that must be considered if security is analysed. In this example, the periodic estimation may leak to the cloud when estimators are within an estimation range or context, and a solution may be sending dummy measurements with $s_{k,i} = \mathcal{E}_{\mathsf{pk}}(\mathsf{E}_{N,\phi}(0))$ when estimator $i$ is out of range. This extension is presented only as an example of when care needs to be taken to maintain desired security goals, but in general, extensions and solutions are task-dependent and not a focus of this work.

### 3.5.2. Simulation

Fusion estimates and error covariances from the proposed encrypted FCI method differ from unencrypted FCI only when quantisation errors are large or summation overflows occur. As stated in section [subsec:encoding], when the Paillier modulus $N$ is large, these errors can often be considered negligible. In this section, we demonstrate this similarity in performance between the encrypted and unencrypted FCI fusion algorithms with a simulation. Code was written in the Python programming language, using the phe Paillier encryption scheme library [PythonPaillier2013] and a 512 bit length key (bit length of $N$). The simulation implements a linear constant velocity model,

$$\underline{x}_k = \begin{bmatrix} 1 & 0.5 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0.5 \\ 0 & 0 & 0 & 0 \end{bmatrix} \cdot \underline{x}_{k-1} + \underline{w}_k \,, \tag{3.24}$$

with noise term $\underline{w}_k \sim \mathcal{N}(\underline{0}, \mathbf{Q})$ and

$$\mathbf{Q} = 10^{-3} \cdot \begin{bmatrix} 0.42 & 1.25 & 0 & 0 \\ 1.25 & 5 & 0 & 0 \\ 0 & 0 & 0.42 & 1.25 \\ 0 & 0 & 1.25 & 5 \end{bmatrix} . \tag{3.25}$$

At each timestep $k$, the system state $\underline{x}_k$ is estimated by $m = 4$ estimators, $1 \leq i \leq 4$, using a standard linear Kalman filter (KF) [haugBayesianEstimationTracking2012] and producing estimates and error covariances $\hat{\underline{x}}_{k,i}$ and $\mathbf{P}_{k,i}$, respectively. The measurements used by the KF, $\underline{z}_{k,i}$, follow the measurement models

$$\underline{z}_{k,i} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \cdot \underline{x}_k + \underline{v}_{k,i} \,, \tag{3.26}$$

with noise terms $\underline{v}_{k,i} \sim \mathcal{N}(\underline{0}, \mathbf{R}_i)$ and covariances sampled indepedently, resulting in

$$\mathbf{R}_1 = \begin{bmatrix} 4.77 & -0.15 \\ -0.15 & 4.94 \end{bmatrix}, \ \mathbf{R}_2 = \begin{bmatrix} 2.99 & -0.55 \\ -0.55 & 4.44 \end{bmatrix}, \\ \mathbf{R}_3 = \begin{bmatrix} 2.06 & 0.68 \\ 0.68 & 1.96 \end{bmatrix} \text{ and } \mathbf{R}_4 = \begin{bmatrix} 1.17 & 0.80 \\ 0.80 & 0.64 \end{bmatrix}. \tag{3.27}$$

The fusion results of 1000 simulation runs are shown in figure 3.5. From the figure, we

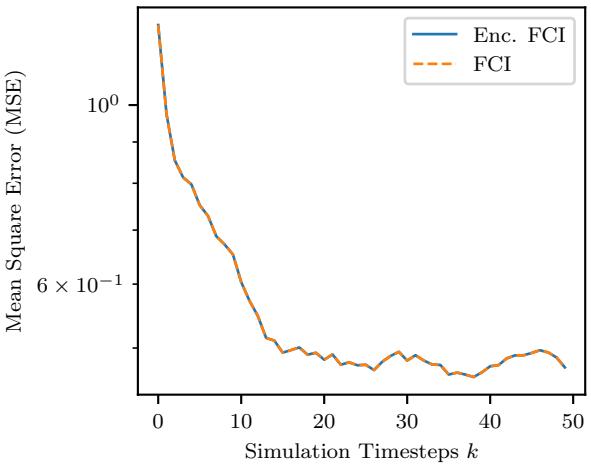



Figure 3.5.: Average RMSE of encrypted and unencrypted FCI fusion over 1000 simulations.

can see the expected similarity in performance between the encrypted and unencrypted FCI methods. Additionally, we note that the current recommended key length for the Paillier encryption scheme is 2048 bits [barkerRecommendationPairWiseKey2019], easily supporting a modulus $N$ and fractional precision $\phi$ that guarantee similar performance.

## 3.6. Conclusions

# 4. Distributed Non-Linear Measurement Fusion with Untrusted Participants

## 4.1. Problem Formulation

In this work, we consider the context of privacy-preserving range sensor navigation, where we want no sensor to learn any information about the navigator or other sensors beyond their local measurements, and the navigator not to learn any information about individual sensors beyond its location estimate. The problem is two-fold, in that we require explicit cryptographic requirements with a suitable encryption scheme meeting them as well as an estimation scheme that can use the encryption in the context of range-only navigation.

To give a formal cryptographic requirement in a distributed setting, we must first consider the communication requirements of our context and define attacker capabilities and the desired security of a suitable encryption scheme. In this section, we will define a communication protocol and the relevant formal definition of security we aim to achieve, followed by the estimation problem to which we will apply it.

### 4.1.1. Formal Cryptographic Problem

The communication between the navigator and sensors in our estimation problem will be decomposed into a simple two-step bi-directional protocol that will simplify defining formal security. In section 4.3.2, we will show how this protocol is sufficient to compute the location estimate at a navigator while meeting our desired privacy goals. The communication protocol is as follows.

At every *instance* $t$ (used to distinguish from an estimation *timestep*), the navigator first broadcasts $m$ weights $\omega_j^{(t)}, j \in \{1, \dots, m\}$ to all sensors $i \in \{1, \dots, n\}$, who individually compute linear combinations $l_i^{(t)} = \sum_{j=1}^{m} a_{j,i}^{(t)} \omega_i^{(t)}$ based on their measurement data $a_{j,i}$. Linear combinations are then sent back to the navigator, who computes their sum $\sum_{i=1}^{n} l_i^{(t)}$. This two-step linear combination aggregation protocol has been visually displayed in figure 4.1. In addition, we note that an alternative approach to the two-step protocol is computing $\sum_{j=1}^{m} (\omega_j^{(t)} \sum_{i=1}^{n} a_{i,j}^{(t)})$ at the navigator, requiring only values $a_{i,j}^{(t)}, j \in \{1, \dots, m\}$ to be sent from each sensor $i$. We justify the use of bi-directional communication by reducing communication costs when the number of weights is larger than the number of sensors, $m > n$, and by sending fewer weights in the presence of repeats, as will be shown to be the case in section 4.3.2.

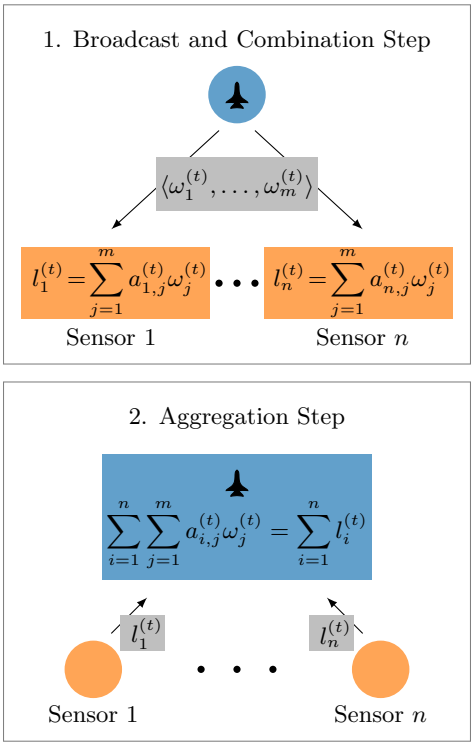



Figure 4.1.: Required linear combination aggregation steps at instance $t$.

Before giving a formal definition for the construction and security of our desired encryption scheme, we make the following assumptions about the capabilities of the participants.

**Global Navigator Broadcast** We assume that broadcast information from the navigator is received by *all* sensors involved in the protocol.

**Consistent Navigator Broadcast** We assume that broadcast information from the navigator is received equally by all sensors. This means the navigator may not send different weights to individual sensors during a single instance $t$.

**Honest-but-Curious Sensors** We adopt the honest-but-curious attacker model for all involved sensors, meaning that they follow the localisation procedure correctly but may store or use any gained sensitive information.

We justify the global broadcast assumption by noting that any subset of sensors within the range of the navigator can be considered a group and treated as the global set during estimation, generalising the method, while the widespread use of cheap non-directional antennas supports the assumption of consistent broadcasts. The final assumption refers to the known problem of misbehaving sensors [lazosSeRLocSecureRangeindependent2004,bengalOutlierDetection2005], often requiring additional complicated detection mechanisms, and will not be considered in this work.

We are now ready to define the type of encryption scheme we want for the specified communication protocol and the security guarantees it should provide. We let a linear combination aggregation scheme be defined as a tuple of the four algorithms $(\mathsf{Setup}, \mathsf{Enc}, \mathsf{CombEnc}, \mathsf{AggDec})$. These will be used by a trusted setup party, the navigator, and sensors $i \in \{1, \ldots, n\}$. They are defined as follows.

$\mathsf{Setup}(\kappa)$ On input of security parameter $\kappa$, generate public parameters $\mathsf{pub}$, the number of weights $m$, the navigator's public and private keys $pk_0$ and $sk_0$ and the sensor private keys $sk_i$, $i \in \{1, \ldots, n\}$.

$\mathsf{Enc}(pk_0, x)$ The navigator and sensors can encrypt any value $x$ with the navigator's public key $pk_0$ and obtain the encryption $\mathcal{E}_{pk_0}(x)$.

$\mathsf{CombEnc}(t, pk_0, sk_i, \mathcal{E}(\omega_1^{(t)}), \ldots, \mathcal{E}(\omega_m^{(t)}), a_{i,1}^{(t)}, \ldots, a_{i,m}^{(t)})$ At instance $t$, sensor $i$ computes and obtains the encrypted linear combination denoted $l_i^{(t)} = \mathcal{E}_{pk_0,sk_i}(\sum_{j=1}^{m} a_{i,j}^{(t)} \omega_j^{(t)})$ using its secret key $sk_i$.

$\mathsf{AggDec}(t, pk_0, sk_0, l_1^{(t)}, \ldots, l_n^{(t)})$ At instance $t$, the navigator computes the aggregation of linear combinations $\sum_{i=1}^{n} l_i^{(t)} = \sum_{i=1}^{n} \sum_{j=1}^{m} a_{i,j}^{(t)} \omega_j^{(t)}$ using its public and private keys $pk_0$, $sk_0$.

The security notions we want these algorithms to meet reflect the previously stated estimation privacy goals. The navigator should learn no information from individual sensors while sensors should learn no information from the navigator or any other sensors. In the context of the introduced communication protocol, this can be summarised as the following notions.

**Indistinguishable Weights** No colluding subset of sensors gains any new knowledge about the navigator weights $\omega_j^{(t)}$, $j \in \{1, \ldots, m\}$ when receiving only their encryptions from the current and previous instances and having the ability to encrypt plaintexts of their choice.

**Linear Combination Aggregator Obliviousness** No colluding subset *excluding* the navigator gains additional information about the remaining sensor values to be weighted, $a_{i,j}^{(t)}$, $j \in \{1, \ldots, m\}$, where sensor $i$ is not colluding, given only encryptions of their linear combinations $l_i$ from the current and previous instances. Any colluding subset *including* the navigator learns only the sum of all linear combinations weighted by weights of their choice, $\sum_{i=1}^{n} l_i^{(t)} = \sum_{i=1}^{n} \sum_{j=1}^{m} a_{i,j}^{(t)} \omega_j^{(t)}$.

While indistinguishable weights can be achieved by encrypting weights with an encryption scheme meeting the notion of Indistinguishability under the Chosen Plaintext Attack (IND-CPA) [katzIntroductionModernCryptography2008], the novel notion of Linear Combination Aggregator Obliviousness (LCAO) has been formalised as a typical cryptographic game between attacker and challenger in the appendix [app:lcao]. Lastly, we conclude the cryptographic problem definition with the following important remark.

*Remark.* A leakage function including weights from the navigator requires extra care to be taken when giving its definition. If an attacker compromises the navigator, they have control over the weights, and therefore the leakage function. We note that in the leakage function above, $\sum_{i=1}^{n} \sum_{j=1}^{m} a_{i,j}^{(t)} \omega_j^{(t)}$, an individual sum weighted by the same weight may be learnt by an attacker, *e.g.*, $\sum_{i=1}^{n} a_{i,1}^{(t)}$ given weights $(1, 0, \ldots, 0)$, but that individual sensor values $a_{i,j}^{(t)}$ remain private due to the assumption of a consistent broadcast.

### 4.1.2. Estimation problem

The estimation problem we consider, for which we will reformulate communication to the protocol above, is localisation with range-only sensors. In this work, we will focus on the two-dimensional case for simplicity but will derive methods suitable for extension to a three-dimensional equivalent. The state that we wish to estimate must capture the navigator position, $x$ and $y$, and may contain any other components relevant to the system. It is of the form

$$\underline{x} = \begin{bmatrix} x & y & \cdots \end{bmatrix}^\top . \tag{4.1}$$

This state evolves following some known system model, which at timestep $k$ can be written as

$$\underline{x}_k = \underline{f}_k(\underline{x}_{k-1}, \underline{w}_k) , \tag{4.2}$$

with noise term $\underline{w}_k$. Measurements of $\underline{x}_k$ follow a measurement model dependent on sensor $i \in \{1, \ldots, n\}$, given by

$$z_{k,i} = h_i(\underline{x}_k) + v_{k,i} , \tag{4.3}$$

with Gaussian measurement noises $v_{k,i} \sim \mathcal{N}(0, r_{k,i})$ and measurement function

$$\begin{aligned} h_i(\underline{x}) &= \left\| \begin{bmatrix} x & y \end{bmatrix}^\top - \underline{s}_i \right\| \\ &= \sqrt{(x - s_{x,i})^2 + (y - s_{y,i})^2}\,, \end{aligned} \tag{4.4}$$

where

$$\underline{s}_i = \begin{bmatrix} s_{x,i} & s_{y,i} \end{bmatrix}^\top \tag{4.5}$$

is the location of sensor $i$.

We aim to provide a filter that estimates the navigator's state $\underline{x}_k$, at every timestep $k$, without learning sensor positions $\underline{s}_i$, measurements $z_{k,i}$ and measurement variances $r_{k,i}$ beyond the information in the corresponding aggregation leakage function. Similarly, sensors should not learn any information about current state estimates or any other sensor information. Leakage will be further discussed in section 4.3.2, but we note that from any sequential state estimates, following known models, some sensor information leakage can be computed by the navigator. In the context of our leakage function, we will show that this corresponds to the global sums of private sensor information, while individual, or subsets of sensors', information remains private. Similarly, corrupted sensors with access to one or more measurements can produce state estimates of their own, leaking information about navigator state estimates, however, the most accurate estimates, requiring all measurements, will always remain private to the navigator.

## 4.2. Related Literature

## 4.3. Confidential Range-Only Localisation

### 4.3.1. Private Linear Combination Aggregation Scheme

In this section, we introduce an encryption scheme meeting the desired security properties in section 4.1.1. The scheme is a combination of the Paillier and Joye-Libert schemes and provides encrypted weights meeting IND-CPA and encrypted aggregation meeting the notion of LCAO defined in section 4.1.1. Similarly to its constituents, the scheme bases its security on the DCRA and, as with the Joye-Libert scheme, requires a trusted party for initial key generation and distribution.

As aggregation is typically performed on scalar inputs, we extend our notation to the context of multidimensional estimation data by letting an instance $t_{k,\tau}$ uniquely capture the scalar aggregation during an estimation timestep $k$ for a single element with position index $\tau$. To achieve this in practice, any injective function can be used, such as the concatenation $t_{k,\tau} = k \parallel \tau$. The four algorithms defining our scheme are given as follows.

$\mathsf{Setup}(\kappa)$ On input parameter $\kappa$, generate two equal-length, sufficiently large, primes $p$ and $q$, and compute $N = pq$. Define a hash function $H : \mathbb{Z} \to \mathbb{Z}^*_{N^2}$, choose the number of weights to combine, $m > 1$, and set public parameter $\mathsf{pub} = H$,

navigator public key $pk_0 = N$ and navigator private key $sk_0 = (p, q)$. Sensor secret keys are generated by choosing $sk_i$, $i \in \{1, \ldots, n-1\}$ uniformly from $\mathbb{Z}_{N^2}$ and setting the last key to $sk_n = -\sum_{i=1}^{n-1} sk_i$.

$\mathsf{Enc}(pk_0, x)$ Public-key encryption is computed by the Paillier encryption scheme with implicit generator $g = N + 1$. This is given by

$$\mathcal{E}_{pk_0}(x) = (N+1)^x \rho^N \pmod{N^2}, \tag{4.6}$$

for a randomly chosen $\rho \in \mathbb{Z}_N$.

$\mathsf{CombEnc}(t_{k,\tau}, pk_0, sk_i, \mathcal{E}_{pk_0}(\omega_1^{(k,\tau)}) \ldots, a_{i,1}^{(k,\tau)} \ldots)$ At the instance $t_{k,\tau}$, encrypted linear combination is given by

$$l_i^{(k,\tau)} = H(t_{k,\tau})^{sk_i} \prod_{j=1}^{m} \mathcal{E}_{pk_0}(\omega_j^{(k,\tau)})^{a_{i,j}^{(k,\tau)}} \pmod{N^2}, \tag{4.7}$$

and makes use of the homomorphic property (2.12). Correctness follows from

$$\begin{aligned}
l_i^{(k,\tau)} &= H(t_{k,\tau})^{sk_i} \prod_{j=1}^{m} \mathcal{E}_{pk_0}(\omega_j^{(k,\tau)})^{a_{i,j}^{(k,\tau)}} \pmod{N^2} \\
&= H(t_{k,\tau})^{sk_i} \prod_{j=1}^{m} \mathcal{E}_{pk_0}(a_{i,j}^{(k,\tau)} \omega_j^{(k,\tau)}) \pmod{N^2} \\
&= H(t_{k,\tau})^{sk_i} \cdot \\
&\qquad \prod_{j=1}^{m} (N+1)^{a_{i,j}^{(k,\tau)} \omega_j^{(k,\tau)}} \rho_j^N \pmod{N^2} \\
&= H(t_{k,\tau})^{sk_i} \cdot \\
&\qquad (N+1)^{\sum_{j=1}^{m} a_{i,j}^{(k,\tau)} \omega_j^{(k,\tau)}} \tilde{\rho}_i^N \pmod{N^2},
\end{aligned}$$

for some values $\rho_j \in \mathbb{Z}_N$, $j \in \{1, \ldots, m\}$ and $\tilde{\rho}_i = \prod_{j=1}^{m} \rho_j$. Here, $\tilde{\rho}_i^N$ and $H(t_{k,\tau})^{sk_i}$ can be considered the noise terms corresponding to the two levels of encryption from $pk_0$ and $sk_i$, respectively.

$\mathsf{AggDec}(t_{k,\tau}, pk_0, sk_0, l_1^{(k,\tau)}, \ldots, l_n^{(k,\tau)})$ Aggregation is computed as $l^{(k,\tau)} = \prod_{i=1}^{n} l_i^{(k,\tau)} \pmod{N^2}$, removing the aggregation noise terms, and is followed by Paillier scheme decryption

$$\begin{aligned}
&\sum_{i=1}^{n} \sum_{j=1}^{m} a_{i,j}^{(k,\tau)} \omega_j^{(k,\tau)} = \\
&\qquad \frac{L((l^{(k,\tau)})^\lambda \pmod{N^2})}{L((N+1)^\lambda \pmod{N^2})} \pmod{N},
\end{aligned} \tag{4.8}$$

with $\lambda = \mathsf{lcm}(p-1, q-1)$ and $L(\psi) = \frac{\psi-1}{N}$. The correctness of the aggregation can be seen from

$$\begin{aligned} l^{(k,\tau)} &= \prod_{i=1}^{n} H(t_{k,\tau})^{sk_i} \cdot \\ &\qquad (N+1)^{\sum_{j=1}^{m} a_{i,j}^{(k,\tau)} \omega_j^{(k,\tau)}} \tilde{\rho}_i^N \pmod{N^2} \\ &= H(t_{k,\tau})^{\sum_{i=1}^{n} sk_i} \cdot \\ &\qquad \prod_{i=1}^{n} (N+1)^{\sum_{j=1}^{m} a_{i,j}^{(k,\tau)} \omega_j^{(k,\tau)}} \tilde{\rho}_i^N \pmod{N^2} \\ &= (N+1)^{\sum_{i=1}^{n} \sum_{j=1}^{m} a_{i,j}^{(k,\tau)} \omega_j^{(k,\tau)}} \tilde{\rho}^N \pmod{N^2}, \end{aligned}$$

for some values $\tilde{\rho}_i \in \mathbb{Z}_N$, $i \in \{1, \ldots, n\}$ and $\tilde{\rho} = \prod_{i=1}^{n} \tilde{\rho}_i$.

Additionally, we note that in the above construction, all weights $\omega_j^{(k,\tau)}$ and values $a_{i,j}^{(k,\tau)}$ are integers and the resulting linear combinations and aggregation are computed modulo $N$.

The security proof of this scheme must both show that encrypted weights meet IND-CPA and that encrypted aggregation meets LCAO. As weights are encrypted with the Paillier encryption scheme, the first requirement is already met. To show that aggregation meets LCAO, a reduction proof is given in the appendix [app:proof].

*Remark.* Given the construction of the scheme above, it can be seen that any weights $\omega_j^{(k,\tau)}$, whose values are known at each sensor, do not need to be broadcast by the navigator. In this case, sensors can replace

$$\mathcal{E}_{pk_0}(\omega_j^{(k,\tau)})^{a_{i,j}^{(k,\tau)}} = (N+1)^{\omega_j^{(k,\tau)} a_{i,j}^{(k,\tau)}} \rho_j^N \pmod{N^2} \tag{4.9}$$

in (4.7), by

$$(N+1)^{\omega_j^{(k,\tau)} a_{i,j}^{(k,\tau)}} \pmod{N^2}. \tag{4.10}$$

This is due to the removal of $\rho_j^N$ terms during decryption and can be used to reduce the navigator's broadcast communication cost by the number of weights $\omega_j^{(k,\tau)}$ that do not hold any information private to the navigator and are known by the sensors in advance.

### 4.3.2. Privacy-Preserving Localisation

With a concrete scheme meeting the LCAO notion, we can now put forward a localisation filter with communication that can be reformulated to the required protocol. To produce an estimate of the state $\underline{x}_k$, we make use of an algebraic reformulation of the Extended Kalman Filter (EKF), the Extended Information Filter (EIF) [maybeckStochasticModelsEstimation1982], which reduces the filter update step to a single summation. The EIF update step requires the predicted state estimate $\underline{\hat{x}}_{k|k-1}$ and estimate covariance

$\mathbf{P}_{k|k-1}$ in the information vector and matrix forms

$$\hat{\underline{y}}_{k|k-1} = \mathbf{P}_{k|k-1}^{-1}\hat{\underline{x}}_{k|k-1} \quad \text{and} \quad \mathbf{Y}_{k|k-1} = \mathbf{P}_{k|k-1}^{-1}\,, \tag{4.11}$$

respectively. In this form, the update equations for $n$ sensor measurements at time $k$, with measurement models (4.3), are given by

$$\begin{aligned}\hat{\underline{y}}_{k|k} &= \hat{\underline{y}}_{k|k-1} + \\ &\sum_{i=1}^{n} \mathbf{H}_{k,i}^{\top} r_i^{-1} \left( z_{k,i} - h_i(\hat{\underline{x}}_{k|k-1}) + \mathbf{H}_{k,i}\hat{\underline{x}}_{k|k-1} \right)\end{aligned} \tag{4.12}$$

and

$$\mathbf{Y}_{k|k} = \mathbf{Y}_{k|k-1} + \sum_{i=1}^{n} \mathbf{H}_{k,i}^{\top} r_i^{-1} \mathbf{H}_{k,i}\,, \tag{4.13}$$

with Jacobians

$$\mathbf{H}_{k,i} = \left.\frac{\partial h_i}{\partial \underline{x}}\right|_{\hat{\underline{x}}_{k|k-1}} \tag{4.14}$$

for sensors $i \in \{1, \ldots, n\}$. After converting the updated information vector and matrix back to state estimate $\hat{\underline{x}}_{k|k}$ and estimate covariance $\mathbf{P}_{k|k}$, the filter's prediction step can be computed by the navigator locally using any suitable filter for the known system model (4.2).

In the form above, at every timestep $k$, all sensitive sensor information required for state estimation is captured in the measurement vector

$$\underline{i}_{k,i} = \mathbf{H}_{k,i}^{\top} r_i^{-1} \left( z_{k,i} - h_i(\hat{\underline{x}}_{k|k-1}) + \mathbf{H}_{k,i}\hat{\underline{x}}_{k|k-1} \right) \tag{4.15}$$

and the measurement matrix

$$\mathbf{I}_{k,i} = \mathbf{H}_{k,i}^{\top} r_i^{-1} \mathbf{H}_{k,i}\,, \tag{4.16}$$

namely, their measurements $z_{k,i}$, measurement variances $r_{k,i}$ and locations $\underline{s}_i$; captured in measurement functions $h_i$ and Jacobians $\mathbf{H}_{k,i}$. To compute $\underline{i}_{k,i}$ and $\mathbf{I}_{k,i}$, however, the current predicted state estimate $\hat{\underline{x}}_{k|k-1}$ is also required (in $h_i$ and $\mathbf{H}_{k,i}$). Therefore, our goal is to rearrange (4.15) and (4.16) as a linear combination of functions of $\hat{\underline{x}}_{k|k-1}$ (the navigator weights) computable at each sensor $i$, to be subsequently aggregated at the navigator. Application of the linear combination aggregation scheme proposed in turn guarantees that sensors do not learn the navigator state, and the navigator learns only the aggregation required for updating its state estimate in (4.12) and (4.13).

### Range Measurement Modification

The first thing we notice when wanting to rearrange $\underline{i}_{k,i}$ and $\mathbf{I}_{k,i}$ to a linear combination of functions of $\hat{\underline{x}}_{k|k-1}$, is that $h_i$ cannot be rearranged in this way due to the present

square-root. Similarly, the Jacobian of $h_i$ at $\underline{\hat{x}}_{k|k-1}$,

$$\mathbf{H}_{k,i} = \begin{bmatrix} \frac{\hat{x}_{k|k-1} - s_{x,i}}{\sqrt{(\hat{x}_{k|k-1} - s_{x,i})^2 + (\hat{y}_{k|k-1} - s_{y,i})^2}} \\ \frac{\hat{y}_{k|k-1} - s_{y,i}}{\sqrt{(\hat{x}_{k|k-1} - s_{x,i})^2 + (\hat{y}_{k|k-1} - s_{y,i})^2}} \\ 0 \\ \vdots \end{bmatrix}^\top , \tag{4.17}$$

cannot be either. We, therefore, consider the modified measurement functions

$$h'_i(\underline{x}) = h_i(\underline{x})^2 . \tag{4.18}$$

A measurement function in this form allows rearrangement of $h'_i$ and the corresponding Jacobian $\mathbf{H}'_{k,i}$ to a linear combination of powers of location elements in $\underline{\hat{x}}_{k|k-1}$, as

$$\begin{aligned} h'_i(\underline{x}) &= \left\| \begin{bmatrix} x & y \end{bmatrix}^\top - \underline{s}_i \right\|^2 \\ &= (x - s_{x,i})^2 + (y - s_{y,i})^2 \\ &= x^2 + y^2 - 2s_{x,i}x - 2s_{y,i}y + s_{x,i}^2 + s_{y,i}^2 \end{aligned} , \tag{4.19}$$

and

$$\mathbf{H}'_{k,i} = \begin{bmatrix} 2\hat{x}_{k|k-1} - 2s_{x,i} \\ 2\hat{y}_{k|k-1} - 2s_{y,i} \\ 0 \\ \vdots \end{bmatrix}^\top . \tag{4.20}$$

Here, $h'_i$ and $\mathbf{H}'_{k,i}$ are linear combinations of $\hat{x}^2_{k|k-1}$, $\hat{y}^2_{k|k-1}$, $\hat{x}_{k|k-1}$ and $\hat{y}_{k|k-1}$. To show how the corresponding modified measurement vectors $\underline{i}'_{k,i}$ and matrices $\mathbf{I}'_{k,i}$ can be similarly rearranged and used for localisation, we also require the existence of measurements following a modified measurement model of the form

$$z'_{k,i} = h'_i(\underline{x}_k) + v'_{k,i} , \tag{4.21}$$

where $z'_{k,i}$ is the modified measurement, and noise term $v'_{k,i}$ is zero-mean and has a known variance $r'_{k,i}$.

Computing $z'_{k,i}$ and its variance $r'_{k,i}$ from the original measurements $z_{k,i}$ are complicated by the noise term $v_{k,i} \sim \mathcal{N}(0, r_{k,i})$, and simply squaring the original range measurements produces

$$\begin{aligned} z^2_{k,i} &= (h_i(\underline{x}_k) + v_{k,i})^2 \\ &= h'_i(\underline{x}_k) + 2h_i(\underline{x}_k)v_{k,i} + v^2_{k,i} , \end{aligned} \tag{4.22}$$

with a new noise term $2h_i(\underline{x}_k)v_{k,i} + v^2_{k,i}$, now dependent on the measurement function $h_i$, and no longer zero-mean. We can compute the mean of this new noise term (a func-

tion of the Gaussian term $v_{k,i}$) as $\mathsf{E}[2h_i(\underline{x}_k)v_{k,i} + v_{k,i}^2] = r_{k,i}$ and mean-adjust modified measurements as

$$\begin{aligned} z'_{k,i} &= z_{k,i}^2 - r_{k,i} \\ &= h_i(\underline{x}_k)^2 + 2h_i(\underline{x}_k)v_{k,i} + v_{k,i}^2 - r_{k,i} \\ &= h'_i(\underline{x}_k) + v'_{k,i}\,, \end{aligned} \tag{4.23}$$

with now zero-mean noise $v'_{k,i} = 2h_i(\underline{x}_k)v_{k,i} + v_{k,i}^2 - r_{k,i}$. The noise in this case (again a function of $v_{k,i}$) has variance

$$\mathsf{Var}[v'_{k,i}] = 4h_i(\underline{x}_k)^2 r_{k,i} + 2r_{k,i}^2 \tag{4.24}$$

and is also dependent on $h_i$. To use the modified measurement (4.23) with the EIF, we require an estimate for $\mathsf{Var}[v'_{k,i}]$ at the sensor as well. Additionally, a conservative estimate (*i.e.*, a larger variance resulting in less confidence in measurements) is desirable to reduce filter divergence. While the naive approach, replacing $h_i(\underline{x}_k)$ with $z_{k,i}$ in (4.24), may not provide a conservative estimate when $z_{k,i}^2 < h_i(\underline{x}_k)^2$, the Gaussianity of $v_{k,i}$ can be exploited to provide a conservative estimate with 95% confidence by shifting the replacement term $z_{k,i}$ by two of its standard deviations $\sqrt{r_{k,i}}$. The modified measurement's variance at timestep $k$ can therefore be conservatively approximated by

$$\begin{aligned} r'_{k,i} &= 4(z_{k,i} + 2\sqrt{r_{k,i}})^2 r_{k,i} + 2r_{k,i}^2 \\ &\gtrapprox \mathsf{Var}[v'_{k,i}]\,, \end{aligned} \tag{4.25}$$

at each sensor $i$.

The modified measurement model (4.21) can now be used for localisation, when measurements are modified by (4.23) and their new variance estimated with (4.25).

### Localisation

To complete the EIF update as a linear combination aggregation, modified vectors $\underline{i}'_{k,i}$ and matrices $\mathbf{I}'_{k,i}$, using the modified measurement model (4.21), can be rearragned as follows.

$$\begin{aligned} \underline{i}'_{k,i} &= \mathbf{H}'^{\top}_{k,i} r'^{-1}_{k,i} (z'_{k,i} - h'_i(\underline{\hat{x}}_{k|k-1}) + \mathbf{H}'_{k,i}\underline{\hat{x}}_{k|k-1}) \\ &= \begin{bmatrix} \alpha_i^{(k,1)} & \alpha_i^{(k,2)} & 0 & \cdots \end{bmatrix}^{\top}, \end{aligned} \tag{4.26}$$

with

$$\begin{aligned}
\alpha_i^{(k,1)} &= (2r'^{-1}_{k,i})\hat{x}^3_{k|k-1} + (2r'^{-1}_{k,i})\hat{x}_{k|k-1}\hat{y}^2_{k|k-1} \\
&\quad + (-2r'^{-1}_{k,i}s_{x,i})\hat{x}^2_{k|k-1} + (-2r'^{-1}_{k,i}s_{x,i})\hat{y}^2_{k|k-1} \\
&\quad + (2r'^{-1}_{k,i}z'_{k,i})\hat{x}_{k|k-1} + (-2r'^{-1}_{k,i}s^2_{x,i})\hat{x}_{k|k-1} \\
&\quad + (-2r'^{-1}_{k,i}s^2_{y,i})\hat{x}_{k|k-1} + (2r'^{-1}_{k,i}s^3_{x,i}) \\
&\quad + (2r'^{-1}_{k,i}s_{x,i}s^2_{y,i}) + (-2r'^{-1}_{k,i}s_{x,i}z'_{k,i}) \text{ and} \\
\alpha_i^{(k,2)} &= (2r'^{-1}_{k,i})\hat{y}^3_{k|k-1} + (2r'^{-1}_{k,i})\hat{x}^2_{k|k-1}\hat{y}_{k|k-1} \\
&\quad + (-2r'^{-1}_{k,i}s_{y,i})\hat{x}^2_{k|k-1} + (-2r'^{-1}_{k,i}s_{y,i})\hat{y}^2_{k|k-1} \\
&\quad + (2r'^{-1}_{k,i}z'_{k,i})\hat{y}_{k|k-1} + (-2r'^{-1}_{k,i}s^2_{x,i})\hat{y}_{k|k-1} \\
&\quad + (-2r'^{-1}_{k,i}s^2_{y,i})\hat{y}_{k|k-1} + (2r'^{-1}_{k,i}s_{y,i}s^2_{x,i}) \\
&\quad + (2r'^{-1}_{k,i}s^3_{y,i}) + (-2r'^{-1}_{k,i}s_{y,i}z'_{k,i}),
\end{aligned}$$

and

$$\begin{aligned}
\mathbf{I}'_{k,i} &= \mathbf{H}'^{\top}_{k,i}r'^{-1}_{k,i}\mathbf{H}'_{k,i} \\
&= \begin{bmatrix} \alpha_i^{(k,3)} & \alpha_i^{(k,4)} & 0 & \cdots \\ \alpha_i^{(k,5)} & \alpha_i^{(k,6)} & 0 & \cdots \\ 0 & 0 & 0 & \cdots \\ \vdots & \vdots & \vdots & \ddots \end{bmatrix},
\end{aligned} \tag{4.27}$$

with

$$\begin{aligned}
\alpha_i^{(k,3)} &= (4r'^{-1}_{k,i})\hat{x}^2_{k|k-1} + (-8r'^{-1}_{k,i}s_{x,i})\hat{x}_{k|k-1} \\
&\quad + (4r'^{-1}_{k,i}s^2_{x,i}), \\
\alpha_i^{(k,4)} &= (4r'^{-1}_{k,i})\hat{x}_{k|k-1}\hat{y}_{k|k-1} + (-4r'^{-1}_{k,i}s_{y,i})\hat{x}_{k|k-1} \\
&\quad + (-4r'^{-1}_{k,i}s_{x,i})\hat{y}_{k|k-1} + (4r'^{-1}_{k,i}s_{x,i}s_{y,i}), \\
\alpha_i^{(k,5)} &= \alpha_i^{(k,4)} \text{ and} \\
\alpha_i^{(k,6)} &= (4r'^{-1}_{k,i})\hat{y}^2_{k|k-1} + (-8r'^{-1}_{k,i}s_{y,i})\hat{y}_{k|k-1} \\
&\quad + (4r'^{-1}_{k,i}s^2_{y,i}).
\end{aligned}$$

The above rearrangements give $\underline{i}'_{k,i}$ and $\mathbf{I}'_{k,i}$ as linear combinations of elements in

$$\begin{aligned}
&\{\hat{x}^3_{k|k-1},\ \hat{y}^3_{k|k-1},\ \hat{x}^2_{k|k-1}\hat{y}_{k|k-1},\ \hat{x}_{k|k-1}\hat{y}^2_{k|k-1}, \\
&\quad \hat{x}^2_{k|k-1},\ \hat{y}^2_{k|k-1},\ \hat{x}_{k|k-1}\hat{y}_{k|k-1},\ \hat{x}_{k|k-1},\ \hat{y}_{k|k-1}\},
\end{aligned} \tag{4.28}$$

which captures all of the private state information in $\underline{\hat{x}}_{k|k-1}$ required at the sensors. The corresponding EIF update steps (4.12) and (4.13) then become

$$\underline{\hat{y}}_{k|k} = \underline{\hat{y}}_{k|k-1} + \sum_{i=1}^{n} \underline{i}'_{k,i} \tag{4.29}$$

and

$$\mathbf{Y}_{k|k} = \mathbf{Y}_{k|k-1} + \sum_{i=1}^{n} \mathbf{I}'_{k,i}\,, \tag{4.30}$$

respectively.

*Remark.* The above has been derived for two-dimensional localisation but can be similarly derived for the three-dimensional case. However, the number of weights increases combinatorially with the number of dimensions, thus affecting the cost of communication as well.

### Pseudocode

Measurement modification, real number encoding and linear combination aggregation are all required to compute the modified EIF from the previous section in a privacy-preserving manner. In this section, we summarise this entire localisation process and give the pseudocode for its execution. For brevity, we will assume $\phi$ and $M = N$ from section [subsec:encoding] to be public information and thus simplify the encoding notation $\mathsf{E}_{N,\phi,d}(\cdot)$ to $\mathsf{E}_d(\cdot)$. The privacy-preserving localisation filter consists of the following steps.

**Setup** The $\mathsf{Setup}$ algorithm from section 4.3.1 is run only once by a trusted party, $N$ and $H$ are made public, and the navigator and sensor secret keys, $sk_0 = \lambda = \mathsf{lcm}(p-1, q-1)$ and $sk_i$, $i \in \{1, \ldots, n\}$, are distributed accordingly.

**Prediction** At each timestep $k$, the navigator computes the prediction of the current state and its covariance with a local filter before encrypting weights (4.28) with algorithm $\mathsf{Enc}$ and broadcasting them to the sensors. This is given by algorithm 4.

**Measurement** At each timestep $k$, sensors modify their measurements with (4.23) and (4.25) before computing encryptions of $\underline{i}'_{k,i}$ and $\mathbf{I}'_{k,i}$ using algorithm $\mathsf{CombEnc}$ for each element and sending them back to the navigator. This is given by algorithm 5.

**Update** At each timestep $k$, the navigator aggregates and decrypts recieved measurement vectors and matrices with algorithm $\mathsf{AggDec}$, before computing the EIF update equations (4.29) and (4.30). This is given by algorithm 6.

Algorithms 4, 5 and 6 have also been summarised graphically in figure 4.2. Here, for brevity, $\mathcal{E}_{pk_0,sk_i}(\cdot)$ and $\mathsf{E}_d(\cdot)$ denote elementwise operations with the same parameters.

**Algorithm 4** Navigator Prediction

**procedure** PREDICTION($\underline{\hat{x}}_{k-1|k-1}$, $\mathbf{P}_{k-1|k-1}$, $N$)
  Compute $\underline{\hat{x}}_{k|k-1}$ with a local filter
  Compute $\mathbf{P}_{k|k-1}$ with a local filter
  Compute $\mathsf{E}_0(\hat{x}^3_{k|k-1})$ by (2.20)
  Compute $\mathcal{E}_{pk_0}(\mathsf{E}_0(\hat{x}^3_{k|k-1}))$ by (4.6)
  Broadcast $\mathcal{E}_{pk_0}(\mathsf{E}_0(\hat{x}^3_{k|k-1}))$ to sensors
  **for** Remaining weights in (4.28) **do**
    Broadcast weight in the form above
  **end for**
  **return** $\underline{\hat{x}}_{k|k-1}, \mathbf{P}_{k|k-1}$
**end procedure**

**Algorithm 5** Measurement at Sensor $i$

**procedure** MEASUREMENT($i$, $s_{x,i}$, $s_{y,i}$, $r_{k,i}$, $N$, $H$)
  Measure $z_{k,i}$
  Compute $z'_{k,i}$ by (4.23)
  Compute $r'_{k,i}$ by (4.25)
  Recieve $\mathcal{E}_{pk_0}(\mathsf{E}_0(\hat{x}^3_{k|k-1}))$
  **for** Remaining weights in (4.28) **do**
    Recieve weight in the form above
  **end for**
  Let $\boldsymbol{\alpha}_i^{(k,\tau)}$ represent the encryption of $\alpha_i^{(k,\tau)}$ in (4.26) and (4.27)
  $\boldsymbol{\alpha}_i^{(k,1)} \leftarrow \mathcal{E}_{pk_0}(\mathsf{E}_0(\hat{x}^3_{k|k-1}))^{\mathsf{E}_0(2r'^{-1}_{k,i})}$.
  $\mathcal{E}_{pk_0}(\mathsf{E}_0(\hat{x}_{k|k-1}\hat{y}^2_{k|k-1}))^{\mathsf{E}_0(2r'^{-1}_{k,i})}$.
  $\mathcal{E}_{pk_0}(\mathsf{E}_0(\hat{x}^2_{k|k-1}))^{\mathsf{E}_0(-2r'^{-1}_{k,i}s_{x,i})}$.
  $\mathcal{E}_{pk_0}(\mathsf{E}_0(\hat{y}^2_{k|k-1}))^{\mathsf{E}_0(-2r'^{-1}_{k,i}s_{x,i})}$.
  $\mathcal{E}_{pk_0}(\mathsf{E}_0(\hat{x}_{k|k-1}))^{\mathsf{E}_0(2r'^{-1}_{k,i}z'_{k,i})}$.
  $\mathcal{E}_{pk_0}(\mathsf{E}_0(\hat{x}_{k|k-1}))^{\mathsf{E}_0(-2r'^{-1}_{k,i}s^2_{x,i})}$.
  $\mathcal{E}_{pk_0}(\mathsf{E}_0(\hat{x}_{k|k-1}))^{\mathsf{E}_0(-2r'^{-1}_{k,i}s^2_{y,i})}$.
  $(N+1)^{\mathsf{E}_1(2r'^{-1}_{k,i}s^3_{x,i})}(N+1)^{\mathsf{E}_1(2r'^{-1}_{k,i}s_{x,i}s^2_{y,i})}$.
  $(N+1)^{\mathsf{E}_1(-2r'^{-1}_{k,i}s_{x,i}z'_{k,i})}H(k \parallel 1) \pmod{N^2}$
  Compute remaining $\boldsymbol{\alpha}_i^{(k,\tau)}$ using (4.26), (4.27), (4.7) and the remark from section 4.3.1 in the form above
  **for** $\tau \leftarrow 1$ to 6 **do**
    Send $\boldsymbol{\alpha}_i^{(k,\tau)}$ to the navigator
  **end for**
**end procedure**

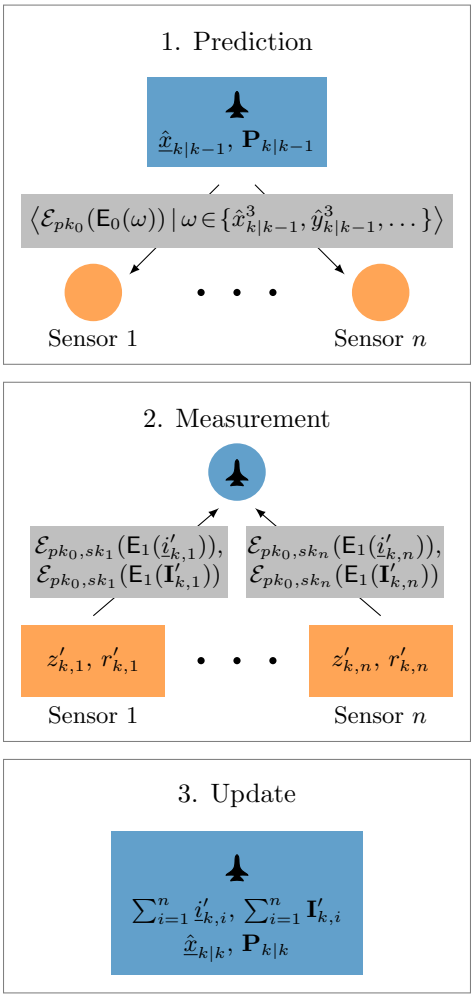



Figure 4.2.: Procedure at timestep $k$ for the proposed privacy-preserving EIF.

**Algorithm 6** Navigator Update

**procedure** UPDATE($\hat{\underline{x}}_{k|k-1}$, $\mathbf{P}_{k|k-1}$, $N$, $\lambda$)
  **for** $\tau \leftarrow 1$ to 6 **do**
    Receive $\boldsymbol{\alpha}_i^{(k,\tau)}$ from each sensor $i \in \{1, \ldots, n\}$
  **end for**
  Let $\boldsymbol{\alpha}^{(k,\tau)}$ represent an encryption of $\sum_{i=1}^{n} \alpha_i^{(k,\tau)}$
  **for** $\tau \leftarrow 1$ to 6 **do**
    $\boldsymbol{\alpha}^{(k,\tau)} \leftarrow \prod_{i=1}^{n} \boldsymbol{\alpha}_i^{(k,\tau)}$
    Compute $\mathcal{D}_{sk_0}(\boldsymbol{\alpha}^{(k,\tau)})$ with $\lambda$ by (4.8)
    Compute $\mathsf{E}_1^{-1}(\mathcal{D}_{sk_0}(\boldsymbol{\alpha}^{(k,\tau)}))$ by (2.21)
  **end for**
  Construct $\sum_{i=1}^{n} \underline{i}'_{k,i}$ and $\sum_{i=1}^{n} \mathbf{I}'_{k,i}$ from decoded decryptions above
  $\hat{\underline{y}}_{k|k} \leftarrow \mathbf{P}_{k|k-1}^{-1} \hat{\underline{x}}_{k|k-1} + \sum_{i=1}^{n} \underline{i}'_{k,i}$
  $\mathbf{Y}_{k|k} \leftarrow \mathbf{P}_{k|k-1}^{-1} + \sum_{i=1}^{n} \mathbf{I}'_{k,i}$
  $\hat{\underline{x}}_{k|k} \leftarrow \mathbf{Y}_{k|k}^{-1} \hat{\underline{y}}_{k|k}$
  $\mathbf{P}_{k|k} \leftarrow \mathbf{Y}_{k|k}^{-1}$
  **return** $\hat{\underline{x}}_{k|k}, \mathbf{P}_{k|k}$
**end procedure**

#### Leakage

With the privacy-preserving EIF defined in the previous section, we can now interpret the aggregation leakage of an LCAO scheme in the context of range sensor localisation. The leakage function from the AggDec algorithm corresponds to the information vector and matrix sums, $\sum_{i=1}^{n} \underline{i}'_{k,i}$ and $\sum_{i=1}^{n} \mathbf{I}'_{k,i}$, respectively. However, recalling that a compromised navigator can learn the individual sums weighted by the same weight, $\{\sum_{i=1}^{n} 2r_{k,i}^{-1},\ \sum_{i=1}^{n} -r_{k,i}^{-1} s_{x,i},\ \sum_{i=1}^{n} -2r_{k,i}^{-1} s_{x,i},\ \ldots\}$ can be leaked as well. From this leakage, we can see that private sensor information, $z'_{k,i}$, $r'_{k,i}$ and $\underline{s}_i$, is present only in their complete sums

$$\sum_{i=1}^{n} z'_{k,i}\,,\ \sum_{i=1}^{n} r'_{k,i}\,,\ \sum_{i=1}^{n} s_{x,i} \text{ and } \sum_{i=1}^{n} s_{y,i}\,, \tag{4.31}$$

which in practice correspond to their averages. Therefore, in the context of our proposed localisation method, LCAO leakage corresponds to the averages of sensor private information, while individual sensor information remains private.

### 4.3.3. Simulation and Results

As well as having shown the theoretical backing for the security of our scheme, we have simulated the proposed localisation method to evaluate its performance. A two-

dimensional, linear, constant velocity process model,

$$\underline{x}_k = \begin{bmatrix} 1 & 0 & 0.5 & 0 \\ 0 & 1 & 0 & 0.5 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \underline{x}_{k-1} + \underline{w}_k \,,$$

where noise term $\underline{w}_k \sim \mathcal{N}(\underline{0}, \mathbf{Q})$ and

$$\mathbf{Q} = \frac{1}{10^3} \cdot \begin{bmatrix} 0.4 & 0 & 1.3 & 0 \\ 0 & 0.4 & 0 & 1.3 \\ 1.3 & 0 & 5.0 & 0 \\ 0 & 1.3 & 0 & 5.0 \end{bmatrix} ,$$

was simulated and tracked with the algorithms in section 4.3.2, using a linear Kalman filter for the navigator's local state prediction. Code was written in the C programming language using the MPI library [theopenmpiprojectOpenMPI2020] to support asynchronous computations by the sensors and the navigator. The MG1 mask generation function and the SHA256 hash function, from the OpenSSL library [theopensslprojectOpenSSL2020], were used to implement the required hash function $H$, and the Libpaillier library [bethencourtLibpaillier2010] was used for the Paillier encryption scheme. Additionally, GNU libraries, GSL [thegsldevelopmentteamGSLGNUScientific2019] and GMP [granlundGMPGNUMultiple2020], were used for algebraic operations and multiple-precision encoded integers, respectively. All execution was performed on a 3.33GHz Xeon W3680 CPU, running on the Windows Subsystem for Linux (WSL).

We have considered multiple sensor layouts, each with four sensors, to capture the dependence of estimated modified measurement variances $r'_{k,i}$ on the original measurements $z_{k,i}$. These layouts of varying sensor distances are shown next to the simulation initial state and a sample track in figure 4.3. To demonstrate the accuracy of the method, we have compared the root mean square error (RMSE) of the privacy-preserving filter to the standard EIF using unmodified measurements, which is algebraically equivalent to the EKF typically used in industry for linearising non-linear state estimation. Estimation in each layout from figure 4.3 consisted of 50 filter iterations and was run 1000 times. Unmodified measurement variances were taken as $r_{k,i} = 5$ for all $k > 0$ and a large fractional precision factor, $\phi = 2^{32}$, was chosen. The results can be seen in figure 4.4. From these results, we can see a strong similarity in filter performance between the privacy-preserving method and that of the traditional EIF. We can also see that the varying average distances between sensors and the navigator have little impact on the differences in performance. We attribute this similarity in RMSE to the conservativeness of estimated modified measurement variances $r'_{k,i}$, eliminating additional filter divergences, and to the high fractional precision factor, keeping computations consistent with the floating-point arithmetic of the EIF.

In addition to filter error, computational performance is important to consider when relying on cryptographic methods. Figure 4.5 shows the averages of 10 execution times

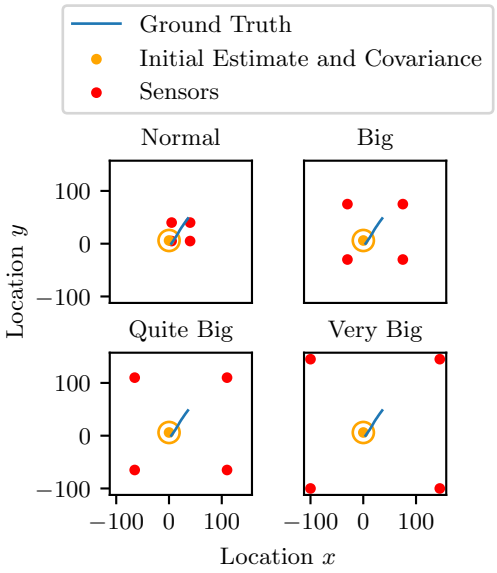



Figure 4.3.: Different simulation layouts with varying distances between navigator and sensors.

when varying the numbers of sensors and key sizes (bit lengths of $N$). Here, increasing the number of sensors primarily affects the number of inter-process communications and aggregation steps due to the asynchronous implementation. We can see that the predominant computational costs stem from cryptographic computations and are directly dependent on the chosen key size. In practice, choosing a key size should take into account the duration of secrecy and the secret key lifetime. When relying on the DCRA for security, the current recommendation for encrypting government documents is the use of 2048 bit length keys [barkerRecommendationPairwiseKey2019]. For our implementation and aforementioned hardware, this results in a filter update roughly every $1.7s$. In a scenario where sensors are mobile and past navigations can be made public, reduced key sizes can be considered, while a further decrease in computation time could be achieved with code optimisations and more powerful hardware.

### 4.3.4. Solvable Sub-Class of Non-Linear Measurement Models

## 4.4. Conclusions

We have presented a localisation filter in the presence of range-only sensors, which preserves both navigator and sensor privacies. A suitable cryptographic scheme has been introduced and a filter implementation compared and evaluated. Privacy-preserving range-only localisation is suitable for use in environments where sensor networks are untrusted or location is considered private and we hope to extend the method to broader measurement models in the future. Additional future work includes exploring more computationally efficient encryption schemes, the security implications of sensors that are

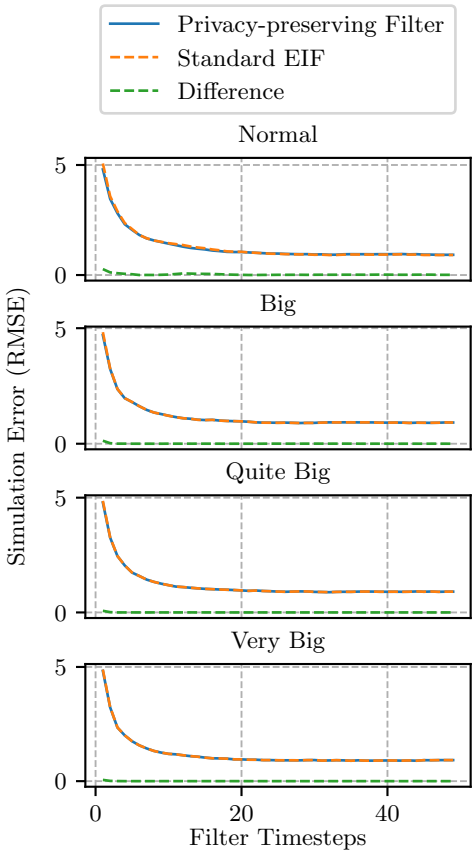



Figure 4.4.: Average RMSE of our privacy-preserving filter and the standard EIF for different layouts.

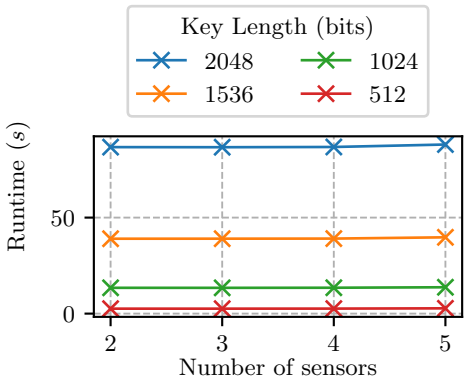



Figure 4.5.: Runtimes for varying key sizes and numbers of sensors.

not only honest-but-curious and expanding the LCAO notion to enforce the consistent broadcast assumption.

# 5. Provable Estimation Performances

## 5.1. Problem Formulation

## 5.2. Related Literature

## 5.3. Covariance Privilege

## 5.4. Privileged Estimation for Linear Systems

### 5.4.1. Extension to Non-Linear Systems

## 5.5. Fusion in Privileged Estimation Environments

## 5.6. Conclusions

# 6. Conclusion

# A. Linear-Combination Aggregator Obliviousness

# B. Cryptographic Proof of LCAO Scheme Security

# List of Figures

# List of Tables

# Bibliography


[1] B. D. O. Anderson and J. B. Moore, *Optimal Filtering*. Dover Publications, 1979.

[2] D. Simon, *Optimal State Estimation: Kalman, H Infinity, and Nonlinear Approaches*. John Wiley & Sons, 2006.

[3] A. G. O. Mutambara, *Decentralized Estimation and Control for Multisensor Systems*. CRC press, 1998.

[4] M. Liggins, C. Y. Chong, D. Hall, and J. Llinas, *Distributed Data Fusion for Network-Centric Operations*. CRC Press, 2012.

[5] C. Y. Chong, "Forty Years of Distributed Estimation: A Review of Noteworthy Developments," in *IEEE ISIF Workshop on Sensor Data Fusion: Trends, Solutions, Applications (SDF 2017)*, 2017, pp. 1–10.

[6] A. J. Haug, *Bayesian Estimation and Tracking: A Practical Guide*. John Wiley & Sons, 2012.

[7] B. Noack, J. Sijs, M. Reinhardt, and U. D. Hanebeck, *Treatment of Dependent Information in Multisensor Kalman Filtering and Data Fusion*. CRC Press, 2017, pp. 169–192.

[8] M. Brenner, J. Wiebelitz, G. von Voigt, and M. Smith, "Secret Program Execution in the Cloud Applying Homomorphic Encryption," in *5th IEEE International Conference on Digital Ecosystems and Technologies (DEST)*, 2011, pp. 114–119.

[9] K. Ren, C. Wang, and Q. Wang, "Security Challenges for the Public Cloud," *IEEE Internet Computing*, vol. 16, no. 1, pp. 69–73, 2012.

[10] S. Gueron, "Intel Advanced Encryption Standard (AES) New Instructions Set," *Intel Corporation*, 2010.

[11] R. L. Rivest, A. Shamir, and L. Adleman, "A Method for Obtaining Digital Signatures and Public-key Cryptosystems," *Communications of the ACM (CACM)*, vol. 21, no. 2, pp. 120–126, 1978.

[12] P. Paillier, "Public-Key Cryptosystems Based on Composite Degree Residuosity Classes," in *Advances in Cryptology (EUROCRYPT)*. Springer, 1999, pp. 223–238.

[13] E. Shi, T.-H. H. Chan, and E. Rieffel, "Privacy-Preserving Aggregation of Time-Series Data," *Annual Network & Distributed System Security Symposium (NDSS)*, p. 17, 2011.

[14] M. Joye and B. Libert, "A Scalable Scheme for Privacy-Preserving Aggregation of Time-Series Data," in *International Conference on Financial Cryptography and Data Security*, ser. Lecture Notes in Computer Science. Springer, 2013, pp. 111–125.

[15] J. Chotard, E. Dufour Sans, R. Gay, D. H. Phan, and D. Pointcheval, "Decentralized Multi-Client Functional Encryption for Inner Product," in *Advances in Cryptology (ASIACRYPT)*, ser. Lecture Notes in Computer Science. Springer, 2018, pp. 703–732.

[16] M. E. Andrés, N. E. Bordenabe, K. Chatzikokolakis, and C. Palamidessi, "Geo-Indistinguishability: Differential Privacy for Location-Based Systems," in *ACM SIGSAC Conference on Computer & Communications Security*, ser. CCS '13. New York, NY, USA: Association for Computing Machinery, 2013, pp. 901–914.