# Localisation and Sensor Privacy Using the Extended Information Filter and Private Linear-Combination Aggregation [⋆]

Marko Ristic [a], Benjamin Noack [a], Uwe D. Hanebeck [a]

[a] *Intelligent Sensor-Actuator-Systems Laboratory, Institute for Anthropomatics, Karlsruhe Institute of Technology, 76131 Karlsruhe, Germany*

**Abstract**

Distributed state estimation and localisation methods have become increasingly popular with the rise of ubiquitous computing, and have led naturally to an increased concern regarding data and estimation privacy. Traditional distributed sensor navigation methods involve the leakage of sensor information or navigator location during localisation protocols and fail to preserve participants' data privacy. Existing approaches which provide such guarantees fail to address sensor and navigator privacy in some common, model-based, non-linear measurement, localisation methods and forfeit broad applicability. We define a cryptographically secure linear-combination aggregation scheme which we apply to the Extended Kalman Filter with range-sensor measurements, and show that navigator location, sensor locations and sensor measurements can remain private during navigation. The security requirements, leakage, and cryptographic proof are given for the private filter and aggregation scheme, and simulations of the filter are used to evaluate the accuracy and performance of the method. Our approach defines a novel, computationally plausible and cryptographically private, model-based localisation filter with direct application to environments where nodes may not be fully trusted and data is considered sensitive.

*Key words:* System state estimation; Data privacy; Sensor fusion; Kalman filters.

## 1 Introduction

Localisation methods in distributed sensor environments have long been an active topic of research []. Ongoing advancements in portable computing power and sensor capabilities have led to the development of various forms of localisation methods, from range-free and range-based signal strength measurements [], echo-based range estimates [] and magnetic field measurements [] to the TDOA and TOA pseudo-range multilateration methods []. Range-based localisation methods, including the use of radar measurements, GPS clocks and signal strength measurements, have found particularly large application due to their robustness and accuracy []. These methods use distance measurements, or computed estimates, to produce location estimates and typically involve gathering sensor details such as measurement noise, geographical location and observations centrally []. Broadcasting or gathering information in this way, however, implies a trust between participants within a trusted communication network to ensure shared data may not be used for individual malicious gain. With the advancements in cloud and ubiquitous computing, changes in possible use-cases involving distributed sensors have made these requirements of information security and user privacy more apparent than before [14,15]. For example, a scenario of aircraft localisation in the presence of privately operated measurement stations may imply measurements and locations of stations should be kept private from navigating vehicles. Alternatively, autonomous vehicle state estimation may benefit from neighbouring vehicles' information while no participant wishes to share information about their hardware. Additionally, the dispatching of any expensive intermediate computations to service-providing cloud operators may require ensuring the privacy of information involved in the computations. All of these scenarios, and the one we will consider, require methods for computing specific operations while ensuring the

[⋆] This paper was not presented at any IFAC meeting. Corresponding author Uwe D. Hanebeck. Tel. +49–721–608–43909.

*Email addresses:* `marko.ristic@kit.edu` (Marko Ristic), `noack@kit.edu` (Benjamin Noack), `uwe.hanebeck@kit.edu` (Uwe D. Hanebeck).

privacy of some or all of the input data. Our contributions in this work will be presented in two parts. We first propose a novel aggregation-based encryption scheme which allows private aggregation of weighted sums, before we develop a localisation filter using this scheme such that sensor and navigator privacy is preserved.

The remainder of the paper is structured as follows. In Section 3 we introduce the localisation problem, the relevant aggregation opertation for its solution and the restrictions on involved parties. Sections 4 and 5 will summarise cryptographic preliminaries and introduce a novel security notion for our required aggregation before defining and proving an encryption scheme which satisfies it. Sections 6 and 7 introduce localisation preliminaries and our privacy-preserving localisation method using the secure aggregation scheme. We show and discuss simulation results in Section 8 and conclude our work in Section 9.

### *1.1 Notation*

Throughout this work, we will use the following notation. Lowercase characters are scalars, $a$, vectors are underlined, $\underline{a}$, and random variables are bold, $\boldsymbol{a}$. Uppercase bold characters are matrices, $\mathbf{A}$, with $\mathbf{A}^{-1}$ denoting the inverse and $\mathbf{A}^\top$ the transpose. $|a|$ is the absolute value of $a$ and $\|\underline{a}\|$ the vector normal. $\mathrm{E}\{\cdot\}$ and $\mathrm{Var}\{\cdot\}$ are used for the expected value and variance, respectively. Encryption and decryption with key $k$ are denoted $\mathcal{E}_k(\cdot)$ and $\mathcal{D}_k(\cdot)$, respectively. $\lfloor\cdot\rceil$ rounds to the nearest integer, $a|b$ stands for "$a$ divides $b$", and $a\|b$ for a bitwise concatenation of $a$ and $b$. Sets will be written as $\{\dots\}$ while ordered sequences as $\langle\dots\rangle$.

## 2 Literature Overview

In this section, we will discuss methods for providing security guarantees in signal processing tasks, and some applications to localisation and estimation problems.

### *2.1 Encryption Schemes in Signal Processing*

Of the methods used to preserve privacy during distributed signal processing, it is common for the information exchanging portion of the algorithm to be reduced to a form of aggregation of involved parties' data []. The most common methods for ensuring privacy during aggregation have been described below.

**Differential Privacy** These techniques formalise preserving the privacy of individual datum within an aggregation operation by introducing known distributions of noise to the resulting aggregation []. In the cases with no trusted aggregator, local differential privacy is achieved by introducing noise to individual inputs [norm, PHE ones]. Although differential privacy has strong computational benefits due to its implementational simplicity, it provides relatively weak security guarantees for multi-party networks, with no guarantees in the presence of multiple colluding corrupted parties. Additionally, noisy aggregation results hinder its use in scenarios where high precision is a desired property such as aircraft localisation.

**Aggregator Oblivious Encryption** This method of encryption formalises the privacy of *all* aggregation data from honest parties []. Encryptions of aggregation data are proven to be computationally indistinguishable and guarantee that only their sum can be learned by an aggregator, whether or not they are trusted. The originally proposed method [] is restricted to a small input domain which has been expanded and computationally simplified in later works [joye-lib,student-one]. The aggregation operation in aggregator obliviousness is, however, strictly limited to the sum operation and restricts its applicability to simple processing tasks such as smart-meter mean calculation [].

In addition to the private aggregation of data, it may also be the case that private data needs to be modified by an external party before being returned, as with privacy-preserving cloud computing. These methods have been broadly grouped into two domains and described below.

**Fully Homomorphic Encryption** These schemes allow all algebraic computations to be performed on encrypted data without the computing party learning any information about inputs, intermediate values or results []. Although theoretically ideal for many privacy-preserving tasks, current implementations are still computationally impractical for real-time or large scale tasks [].

**Partially Homomorphic Encryption** A simplification of fully homomorphic schemes, these schemes provide only a subset, typically one, of algebraic operations on encrypted data []. The reduced computational requirements and implementation simplicity have made partially homomorphic encryption schemes commonly used in privacy-preserving signal processing tasks, including private matrix multiplication [], set intersection computation [] and control input aggregation []. However, these works are relatively restricted in application due to the limited operations provided.

Lastly, recent developments in encryption schemes have provided some novel methods also suitable for privacy-preserving computation in distributed environments.

**Multi-Client Function Encryption** This type of encryption allows the computing of a specific function given encrypted inputs []. It is suitable for privacy-preserving distributed data as encryptions must be combined before a result is obtained, and can be considered a generalisation of the aggregation schemes described earlier. While such schemes that can compute arbitrary functions are still too computationally expensive for practical use [], partial schemes supporting only some functions, such as computing vector dot products [], exist, but have yet to find applications in more complicated signal processing tasks.

Of the aforementioned encryption schemes, some have found uses in privacy-preserving localisation and estimation tasks and we summarise these in the next section.

### *2.2 Encrypted Localisation and Estimation*

Localisation methods exist for a variety of purposes requiring different levels of accuracy and supporting various forms of sensors []. We aim to provide an accurate localisation filter in the presence of range-only sensors.

Relevant estimation methods include both model-based Bayesian estimation methods which make assumptions about target trajectories to form their location estimates, typically based on the Kalman Filter [], and model-free methods where location estimates are made independently of each other [proloc,serloc]. The work in [proloc] uses partially homomorphic encryption to compute model-free location estimates while preserving the privacy of sensor measurements and locations. The work proposes two approaches, using either polygon intersections suffering from geometric dilution of precision, or alternating projection requiring repeated interactive comparison protocols. Neither method considers measurement uncertainty, and no estimate errors are provided.

Model-based approaches can achieve better accuracy due to their exploitation of process dynamic knowledge. In [aristov], partially homomorphic encryption is used to encrypt measurement information and produce location estimates at a navigator. While producing estimate errors and requiring only uni-directional communication, this method exclusively supports linear measurement models and is therefore not suitable for range-only measurements. Model-based encrypted control [farohki summary] also introduces some privacy-preserving methods relevant to estimation. [alexandru] proposes secure distributed control input aggregation using a novel weighted sum aggregation scheme, but requires repeated key distribution among all involved parties. In [farohki], a secure cloud-based control aggregator using partially homomorphic encryption is proposed, but is suitable only for untrusted service-providing cloud architectures.

While we primarily focus on applying encryption schemes to localisation algorithms, homomorphic encryption schemes also require suitable number encoding when used with real numbers. As is the case with localisation sensors, real number measurements need to be encoded as integers with operations consistent under scheme-provided homomorphic operations. The Google BigNum library [google bignum] provides an encoding which supports both addition and unencrypted multiplication when used with an additively homomorphic encryption scheme. It, however, leaks the exponent information of encrypted numbers. In [farohki], a simpler alternative for encoding is proposed, which leaks no information about encrypted numbers, but allows only a single unencrypted multiplication between encodings.

## 3 Problem Statement

The localisation scenario we consider in this work is that of model-based self-navigation using range-only sensors. We consider localisation in the two-dimensional case for simplicity but will derive methods suitable for an extension to the three-dimensional equivalent.

The navigator state is defined as

$$\underline{\boldsymbol{x}} = \begin{bmatrix} \boldsymbol{x} & \boldsymbol{dx} & \boldsymbol{y} & \boldsymbol{dy} \end{bmatrix}^\top . \tag{1}$$

A known process model is followed, which at time $k$ is given by

$$\underline{\boldsymbol{x}}_k = \underline{f}(\underline{\boldsymbol{x}}_{k-1}) + \underline{\boldsymbol{w}}_k , \tag{2}$$

with zero-mean Gaussian process noise $w_k \sim \mathcal{N}(\underline{0}, \mathbf{Q})$. The measurement model is dependant on sensor $i$ and given by

$$\boldsymbol{z}_{k,i} = h_i(\underline{\boldsymbol{x}}_k) + \boldsymbol{v}_{k,i} , \tag{3}$$

with noise $\boldsymbol{v}_{k,i} \sim \mathcal{N}(0, r_i)$, while the measurement function $h_i$, for sensor $i$ at location

$$\underline{s}_i = \begin{bmatrix} s_{x,i} & s_{y,i} \end{bmatrix}^\top , \tag{4}$$

is defined as

$$\begin{aligned} h_i(\underline{\boldsymbol{x}}_k) &= \left\| \begin{bmatrix} \boldsymbol{x}_k & \boldsymbol{y}_k \end{bmatrix}^\top - \underline{s}_i \right\| \\ &= \sqrt{(\boldsymbol{x}_k - s_{x,i})^2 + (\boldsymbol{y}_k - s_{y,i})^2} . \end{aligned} \tag{5}$$

We wish to run a state estimation filter with models (2) and (3) such that all involved sensors $1 \leq i \leq n$ do not learn the estimated state $\underline{\boldsymbol{x}}_k$ and the navigator does not learn sensor locations $\underline{s}_i$, $1 \leq i \leq n$ or their measurements $\boldsymbol{z}_k$ at any time $k$. We motivate these goals with an example. When considering aircraft navigation in the presence of privately-owned range-measuring towers, it is reasonable to assume that the current state of an aircraft may not wish to be disclosed to unknown tower-owning parties. Similarly, tower locations may wish to be kept private from unidentified navigating aircraft. However, the additional safety to passengers, provided by accurate aircraft localisation, may be a goal of all those involved.

### 3.1 Problem Decomposition

The recursive nature of the problem definition above complicates defining and proving of security requirements for the localisation scenario. As with existing estimation methods [] we will reduce the communication requirements, of sensors and the navigator in our case, to an aggregation operation which will make security more easily definable. To support multiple aggregations at each time step $k$, as is required for the aggregation of multi-dimensional data, aggregation instances have been denoted by $t$. The bi-direction communication protocol at instance $t$ consists of of the computation of linear combinations of navigator weights by sensors, before their privacy-preserving aggregation by the navigator. That is, $m$ weights $\omega_j^{(t)}, 1 \leq j \leq m$ are broadcast by the navigator, linear combinations $y_i^{(t)} = \sum_{j=1}^{m} x_{j,i}^{(t)} \omega_i^{(t)}$ are computed by each sensor $1 \leq i \leq n$, and aggregation is computed back at the navigator. This has been summarised in Figure 1.

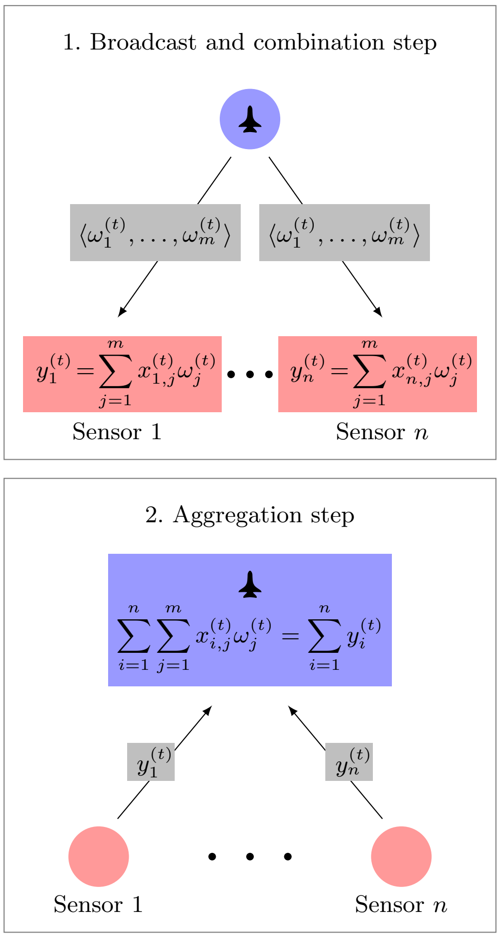
1. Broadcast and combination step
$\langle \omega_1^{(t)}, \dots, \omega_m^{(t)} \rangle$
$\langle \omega_1^{(t)}, \dots, \omega_m^{(t)} \rangle$
$y_1^{(t)} = \sum_{j=1}^{m} x_{1,j}^{(t)} \omega_j^{(t)}$
$y_n^{(t)} = \sum_{j=1}^{m} x_{n,j}^{(t)} \omega_j^{(t)}$
Sensor 1
Sensor $n$
2. Aggregation step
$\sum_{i=1}^{n} \sum_{j=1}^{m} x_{i,j}^{(t)} \omega_j^{(t)} = \sum_{i=1}^{n} y_i^{(t)}$
$y_1^{(t)}$
$y_n^{(t)}$
Sensor 1
Sensor $n$

Fig. 1. Required linear-combination aggregation steps at instance $t$.

In Section 7 we will show how this protocol is sufficient to compute the required estimation information at the sensors required to update the navigator's estimation filter.

**Remark 1** *From the aggregation definition above and the distributive property of real numbers, it is clear that an alternative method to the bi-directional communication requirement is computing* $\sum_{j=1}^{m} \omega_j^{(t)} \sum_{i=1}^{n} x_{i,j}^{(t)}$ *by the navigator, requiring only uni-directional communication by sending values* $x_{i,j}^{(t)}, 1 \leq j \leq m$ *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 7.4.*

### 3.2 Participant Capabilities

Before we can define concrete security requirements and a filtering protocol, we require some assumptions on the capabilities of the navigator and sensors in our problem scenario.

**Global navigator broadcast** Due to sensor location privacy and model non-linearity, bi-directional communication between navigator and sensors is required. We make the assumption that broadcast information from the navigator is received by *all* sensors involved in the protocol.

A problem caused by this requirement is apparent in the context of aircraft navigation. It may be the case that some sensors are not within the range of a navigator broadcast. We loosely propose that this scenario can be handled by pre-defined sensor subsets during an initial step by a trusted party. This has been captured in Figure 2.

**Consistent navigator broadcast** Further, we make the assumption that broadcast information from the navigator is received equal by all sensors. This means the navigator may not send different information to individual sensors during a single timestep.

In the context of wireless communication, the wide-spread use of cheap non-directional antennas make this assumption reasonable.

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

Misbehaving sensors are a known problem in estimation theory, often requiring inconsistency or outlier detection mechanisms. As privacy-preserving implementations of such mechanisms require additional complications, we do not consider the scenario in this work.

**Computational capabilities** The computational requirements for computing encryptions and homomorphic operations are typically more costly than their plaintext equivalents. We make the implicit assumption that all involved parties are computationally capable of computing the required encryption and filter operations.

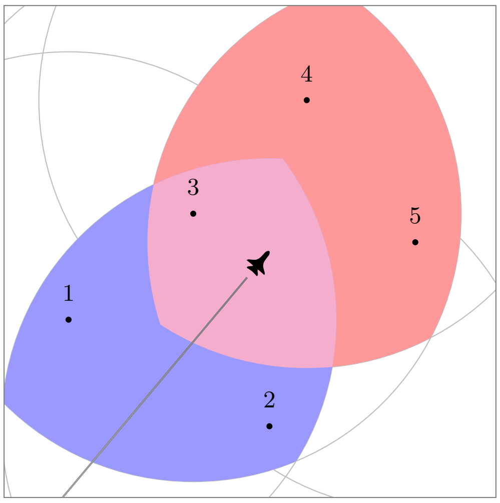

Fig. 2. An example of sensor subsets; $\{1, 2, 3\}$ and $\{3, 4, 5\}$, and the areas where the navigator is in range of all sensors within a subset.

### *3.3 Inherent Leakage*

Leakage of information will be discussed in more detail in Section 7.5, however, we stress that in any estimation problem where process and measurement models are known, knowledge of sequential state estimates naturally leads to the leakage of some measurement information. For this reason, we accept that average information about sensors may be leaked at each time step and instead show that individual sensor information remains private.

## 4 Cryptography Preliminaries

When defining our aggregation security requirements and encryption scheme, we will reference some existing cryptographic security notions, the additively homomorphic Paillier encryption scheme, and the Joye-Libert private aggregation scheme.

### *4.1 Security Notions*

The security of a cryptographic scheme is typically defined by a security *game*, which captures both the desired privacy guarantees, as well as the capabilities of attackers [10]. The typical security notion for a homomorphic encryption scheme is Indistinguishability under Chosen Plaintext Attack (IND-CPA) [7].

**Definition 2** *An encryption scheme meets IND-CPA security if an attacker who can choose plaintext messages to be encrypted at will, gains no additional information about an unknown plaintext message when they learn only its encryption.*

*The formal security game for IND-CPA has been given in Appendix A.*

Private aggregation schemes aim for the security notion of Aggregator Obliviousness (AO) [15].

**Definition 3** *An encryption scheme meets AO security if no colluding subset of participants excluding the aggregator gains additional information about the remaining aggregation values given only their encryptions, while any colluding subset including the aggregator learns only their sum.*

*The formal security game for AO has been given in Appendix B.*

### *4.2 Paillier Encryption Scheme*

The Paillier encryption scheme [13] is an additively homomorphic encryption scheme which 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 [10]. The public key is defined by $(N, g)$ and secret key by $(p, q)$.

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

$$c = g^m r^N \pmod{N^2} \tag{6}$$

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

$$m = \frac{L(c^\lambda \pmod{N^2})}{L(g^\lambda \pmod{N^2})} \pmod{N} \tag{7}$$

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

In addition to encryption and decryption, the following homomorphic functions are provided by the Paillier scheme. $\forall m_1, m_2 \in \mathbb{Z}_N$,

$$\mathcal{D}(\mathcal{E}(m_1)\mathcal{E}(m_2) \pmod{N^2}) = m_1 + m_2 \pmod{N} \tag{8}$$
$$\mathcal{D}(\mathcal{E}(m_1)g^{m_2} \pmod{N^2}) = m_1 + m_2 \pmod{N} \tag{9}$$
$$\mathcal{D}(\mathcal{E}(m_1)^{m_2} \pmod{N^2}) = m_1 m_2 \pmod{N}\,. \tag{10}$$

### *4.3 Joye-Libert Private Aggregation Scheme*

The Joye-Libert private aggregation scheme [9] is a scheme defined on time-series data and meets the security notion of AO. Similarly to the Paillier scheme, it bases its security on the DCRA. A notable difference to a public-key encryption scheme is the need for a trusted party to perform an initial key generation and distribution step.

Key generation is computed by choosing two equal length and sufficiently large primes $p$ and $q$, and computing $N = pq$. Additionally, 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 $sk_i$, $1 \leq i \leq n$ uniformly from $\mathbb{Z}_{N^2}$ and distributing them to all users, while the last key is set as

$$sk_0 = -\sum_{i=1}^{n} sk_i \pmod{N^2},$$

and sent to the aggregator.

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

$$c_i^{(t)} = (N+1)^{m_i^{(t)}} H(t)^{sk_i} \pmod{N^2}, \qquad (11)$$

where the $H(t)^{sk_i}$ can be considered the noise term which hides the again easily computable discrete logarithm $g^{m_i^{(t)}} \pmod{N^2}$, where $g = N + 1$.

When all encryptions $c_i^{(t)}$, $1 \leq i \leq n$ are sent to the aggregator, private summation and decryption are computed by the functions

$$c^{(t)} = H(t)^{sk_0} \prod_{i=1}^{n} c_i^{(t)} \pmod{N^2} \qquad (12)$$

and

$$\sum_{i=1}^{n} m_i^{(t)} = \frac{c^{(t)} - 1}{N}. \qquad (13)$$

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

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

removing all noise terms.

## 5 Private Linear-Combination Aggregation

As stated in Section 3.1, we decompose the localisation method such that the multi-party aggregation protocol, given in Figure 1, captures all communication between the navigator and sensors. To perform this protocol in a secure manner, we must first describe the security properties we aim to achieve during aggregation. Broadly speaking, we want a cryptographic scheme which allows instances of homomorphically computed linear combinations of encrypted weights, to be summed by a private aggregation scheme. That is, we do not want sensors to learn the navigator weights, while we do not want the navigator to learn individual weighted linear combinations. This can be summarised by the two informal security notions:

**Indistinguishable Weights** No colluding subset of sensors gains any additional knowledge about the navigator weights $\omega_j$, $1 \leq j \leq m$ from receiving only their encryptions from the current and previous instances, and the ability to encrypt plaintexts of their choice.

**Private Linear Combination Aggregation** No colluding subset *excluding* the aggregator gains additional information about the remaining sensor values to be weighted $x_{i,j}^{(t)}$, $0 \leq j \leq m$, where sensor $i$ is not colluding, given only encryptions of their linear combinations $y_i$ from the current and previous instances. Any colluding subset *including* the aggregator learns only the sum of all linear combinations weighted by weights of their choice, $\sum_{i=n}^{n} \sum_{j=1}^{m} x_{i,j}^{(t)} \omega_j^{(t)}$.

**Remark 4** *The notion of a leakage function including parameters from the aggregator requires extra care to be taken when giving its definition. Since an attacker may compromise the aggregator, they have control over the choice of these parameters, and therefore over the leakage function. We note that in the leakage function above,* $\sum_{i=n}^{n} \sum_{j=1}^{m} x_{i,j}^{(t)} \omega_j^{(t)}$*, an individual sum weighted by the same weight may be learned by the colluding subset, e.g.* $\sum_{j=1}^{m} x_{1,j}^{(t)}$ *given weights* $(1, 0, \ldots, 0)$*, but that individual sensor values* $x_{i,j}^{(t)}$ *remain private due to the requirement that all sensors receive the same weights.*

From the informal definitions above, it is clear that weights encrypted by an IND-CPA secure encryption scheme are sufficient for the first requirement, while a scheme satisfying AO is not sufficient for the second. To formalise the second requirement, we define a novel encryption type "Linear-Combination Aggregator Oblivious Encryption" and an accompanying security game, which capture the additional weights and modified leakage of AO.

### *5.1 Linear-Combination Aggregator Oblivious Encryption*

We let a linear-combination aggregator oblivious encryption scheme be defined as a tuple of the four algorithms $(\mathsf{Setup}, \mathsf{Enc}, \mathsf{CombEnc}, \mathsf{AggDec})$, defined as

$\mathsf{Setup}(\kappa)$ On input of security paramater $\kappa$, generate public parameters $\mathsf{pub}$, number of weights $m$, the aggregator's public and private keys $pk_0$ and $sk_0$, and the remaining user private keys $sk_i$, $1 \leq i \leq n$.

$\mathsf{Enc}(pk_0, \omega)$ The aggregator and users can encrypt a weight $\omega$ with the aggregator public key $pk_0$, and obtain the encryption $\mathcal{E}_{pk_0}(\omega)$.

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

$\mathsf{AggDec}(t, pk_0, sk_0, y_1^{(t)}, \ldots, y_n^{(t)})$ At time $t$, the aggregator computes the aggregation of linear combinations $\sum_{i=1}^n \sum_{j=1}^m x_{i,j}^{(t)} \omega_j^{(t)}$ using its public and private keys $pk_0$, $sk_0$.

Next, we formalise the security notion of Linear-Combination Aggregator Obliviousness (LCAO) as the following game between attacker and challenger:

**Setup** The challenger runs the $\mathsf{Setup}$ algorithm and gives $\mathsf{pub}$, $m$ and $pk_0$ to the attacker

**Queries** The attacker can now perform encryptions or submit queries that are answered by the challenger. The types of actions are:

(1) *Encryption:* The attacker chooses a weight $\omega$ and computes an encryption of $\omega$ under the aggregator's public key $pk_0$, obtaining $\mathcal{E}_{pk_0}(\omega)$.

(2) *Weight Queries:* The attacker chooses a time $t$ and receives the weights for that time encrypted with the aggregator's public key, $\mathcal{E}_{pk_0}(\omega_j^{(t)})$, $1 \leq j \leq m$.

(3) *Combine Queries:* The attacker chooses a tuple $(i, t, x_{i,1}^{(t)}, \ldots, x_{i,m}^{(t)})$ such that for any two chosen combine query tuples $(i, t, x_{i,1}^{(t)}, \ldots, x_{i,m}^{(t)})$ and $(i', t', x_{i',1}^{\prime(t')}, \ldots, x_{i',m}^{\prime(t')})$, the following condition holds:
$$i = i' \wedge t = t' \implies x_{i,m}^{(t)} = x_{i',m}^{\prime(t')}, \, 1 \leq j \leq m \, .$$
They are given back the encryption of the linear combination $\mathcal{E}_{pk_0, sk_i}(\sum_{j=1}^m x_{i,j}^{(t)} \omega_j^{(t)})$ encrypted under both the aggregator public key $pk_0$ and the secret key $sk_i$.

(4) *Compromise queries:* The attacker chooses $i$ and receives the secret key $sk_i$. The aggregator's secret key may also be compromised (when choosing $i = 0$).

**Challenge** Next, the attacker chooses a time $t^*$, and a subset of users $S \subseteq U$ where $U$ is the complete set of users for which no combine queries, for time $t^*$, and no compromise queries, are made for the duration of the game. The attacker then chooses two series of tuples
$$\langle (i, t^*, x_{i,1}^{(t^*)(0)}, \ldots, x_{i,m}^{(t^*)(0)}) \mid i \in S \rangle$$
and
$$\langle (i, t^*, x_{i,1}^{(t^*)(1)}, \ldots, x_{i,m}^{(t^*)(1)}) \mid i \in S \rangle \, ,$$
and gives them to the challenger. In the case that $0 \in S$ (*i.e.* the aggregator is compromised) and $S = U$, it is additionally required that
$$\sum_{i \in S} \sum_{j=1}^m x_{i,j}^{(t^*)(0)} \omega_j^{(t^*)} = \sum_{i \in S} \sum_{j=1}^m x_{i,j}^{(t^*)(1)} \omega_j^{(t^*)} \, ,$$
for weights $\omega_j^{(t^*)}$, $1 \leq j \leq m$ returned by a *Weight Query* with chosen time $t^*$. The challenger then chooses a random bit $b \in \{1, 0\}$ and returns encryptions
$$\langle \mathcal{E}_{pk_0, sk_i}(\sum_{j=1}^m x_{i,j}^{(t^*)(b)} \omega_j^{(t^*)}) \mid i \in S \rangle \, .$$

**More Queries** The attacker can now perform more encryptions and submit queries, so long as the queries do not break the requirements in the Challenge stage. That is, $S \subseteq U$.

**Guess** At the end, the attacker outputs a bit $b'$ and wins the game only if $b' = b$. The advantage of an attacker $\mathcal{A}$ is defined as
$$\mathsf{Adv}^{LCAO}(\mathcal{A}) := \left| \mathrm{P}[b' = b] - \frac{1}{2} \right| \, .$$

**Definition 5** *An encryption scheme meets LCAO security if no adversary, running in probabilistic-time with respect to security parameter, has more than a negligible advantage in winning the above security game. Probabilities are taken over randomness introduced by $\mathcal{A}$, and in* $\mathsf{Setup}$*,* $\mathsf{Enc}$ *and* $\mathsf{CombEnc}$.

In the next section, we will give a solution to an encryption scheme meeting LCAO security, with IND-CPA secure weight encryption, and give a cryptographic proof for its security.

### *5.2 Our Scheme*

Our scheme is based on the Paillier and Joye-Libert schemes introduced in Section 4, and similarly bases its security on the DCRA. As with Joye-Libert's private aggregation scheme, a trusted party is required for the initial distribution of user secret keys. Below, we give definitions for the four algorithms comprising the linear-combination aggregation encryption scheme.

$\mathsf{Setup}(\kappa)$ 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 an $m > 1$ as the number of weights to combine, and set public parameter $\mathsf{pub} = H$, aggregator public key $pk_0 = N$ and aggregator private key $sk_0 = (p, q)$. The remaining user secret keys are generated by choosing $sk_i$, $1 \leq i \leq n-1$ uniformly from $\mathbb{Z}_{N^2}$ and setting the last key as $sk_n = -\sum_{i=1}^{n-1} sk_i \pmod{N^2}$.

$\mathsf{Enc}(pk_0, \omega)$ Encryption of weights is computed as a Paillier encryption with implicit generator $g = N+1$. This is given by

$$\mathcal{E}_{pk_0}(\omega) = (N+1)^{\omega} r^N \pmod{N^2}\,, \tag{14}$$

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

$\mathsf{CombEnc}(t, pk_0, sk_i, \mathcal{E}_{pk_0}(\omega_1^{(t)}), \ldots, \mathcal{E}_{pk_0}(\omega_m^{(t)}), x_{i,1}^{(t)}, \ldots, x_{i,m}^{(t)})$ The linear combination encryption step at time $t$ is computed as

$$y_i^{(t)} = H(t)^{sk_i} \prod_{j=1}^{m} \mathcal{E}_{pk_0}(\omega_j^{(t)})^{x_{i,j}^{(t)}} \pmod{N^2}\,, \tag{15}$$

and makes use of the homomorphic property (10). Correctness follows from

$$\begin{aligned} y_i^{(t)} &= H(t)^{sk_i} \prod_{j=1}^{m} \mathcal{E}_{pk_0}(\omega_j^{(t)})^{x_{i,j}^{(t)}} \pmod{N^2} \\ &= H(t)^{sk_i} \prod_{j=1}^{m} \mathcal{E}_{pk_0}(x_{i,j}^{(t)}\omega_j^{(t)}) \pmod{N^2} \\ &= H(t)^{sk_i} \prod_{j=1}^{m} (N+1)^{x_{i,j}^{(t)}\omega_j^{(t)}} r_j^N \pmod{N^2} \\ &= H(t)^{sk_i} (N+1)^{\sum_{j=1}^{m} x_{i,j}^{(t)}\omega_j^{(t)}} r_i^N \pmod{N^2}\,, \end{aligned}$$

for some values $r_i, r_j \in \mathbb{Z}_N\,, 1 \leq j \leq m$. Here, $r_i^N$ and $H(t)^{sk_i}$ can be considered the noise terms corresponding to the two levels of encryption from $pk_0$ and $sk_i$, respectively.

$\mathsf{AggDec}(t, pk_0, sk_0, y_1^{(t)}, \ldots, y_n^{(t)})$ Aggregation is computed as $y^{(t)} = \prod_{i=1}^{n} y_i^{(t)} \pmod{N^2}$, removing aggregation noise terms, and is followed by Paillier decryption

$$\sum_{i=1}^{n}\sum_{j=1}^{m} x_{i,j}^{(t)}\omega_j^{(t)} = \frac{L((y^{(t)})^{\lambda} \pmod{N^2})}{L((N+1)^{\lambda} \pmod{N^2})} \pmod{N}\,. \tag{16}$$

The correctness of aggregation can be seen from

$$\begin{aligned} y^{(t)} &= \prod_{i=1}^{n} H(t)^{sk_i} (N+1)^{\sum_{j=1}^{m} x_{i,j}\omega_j} r_i^N \pmod{N^2} \\ &= H(t)^{\sum_{i=1}^{n} sk_i} \cdot \\ &\qquad \prod_{i=1}^{n} (N+1)^{\sum_{j=1}^{m} x_{i,j}^{(t)}\omega_j^{(t)}} r_i^N \pmod{N^2} \\ &= (N+1)^{\sum_{i=1}^{n}\sum_{j=1}^{m} x_{i,j}^{(t)}\omega_j^{(t)}} r'^N \pmod{N^2}\,, \end{aligned}$$

for some values $r_i, r' \in \mathbb{Z}_N$, $1 \leq i \leq n$.

Additionally, we note that in the above construction, all weights $\omega_j^{(t)}$ and values $x_{i,j}^{(t)}$ are integers, and that resulting linear combinations and summations are computed $\pmod{N}$.

### *5.3 Security Proof*

To prove the security of our introduced scheme, we recall the desired security properties of an LCAO secure scheme with IND-CPA secure encrypted weights. From the definition above, weights encrypted with public key $pk_0$ are identical to encryptions of the Paillier scheme, and therefore meet security notion IND-CPA. We omit this proof here and refer readers to the security proof of the Paillier encryption scheme [13] instead.

To show our scheme meets the security notion of LCAO, we prove by contrapositive that for an adversary $\mathcal{A}$ playing against a challenger using *our scheme*, we can create an adversary $\mathcal{A}'$ playing against a challenger $\mathcal{C}$ using the *Joye-Libert scheme*, such that

$$\mathsf{Adv}^{LCAO}(\mathcal{A}) > \eta_1(\kappa) \implies \mathsf{Adv}^{AO}(\mathcal{A}') > \eta_2(\kappa)\,,$$

for some negligible functions $\eta_1$ and $\eta_2$. (*i.e.* if we assume our scheme is not LCAO secure, then the Joye-Libert scheme is not AO secure.) Given the Joye-Libert AO proof in [9], we know our scheme must be LCAO secure and will thus conclude our proof. The proof overview has been given in Figure 3. Additionally, the function $H$ used by our scheme is treated as a *random oracle* in the Joye-Libert AO proof and will, therefore, prove our scheme secure in the random oracle model as well.

**PROOF.** Consider adversary $\mathcal{A}$ playing the LCAO game defined in Section 5.1. The following is a construction of an adversary $\mathcal{A}'$ playing the AO game in Appendix B against a challenger $\mathcal{C}$ using the Joye-Libert aggregation scheme from Section 4.3.

**Setup** When receiving $N$ and $H$ as public parameters from $\mathcal{C}$, choose an $m > 1$ and give public parameter $H$, number of weights $m$, and $pk_0 = N$ to $\mathcal{A}$.

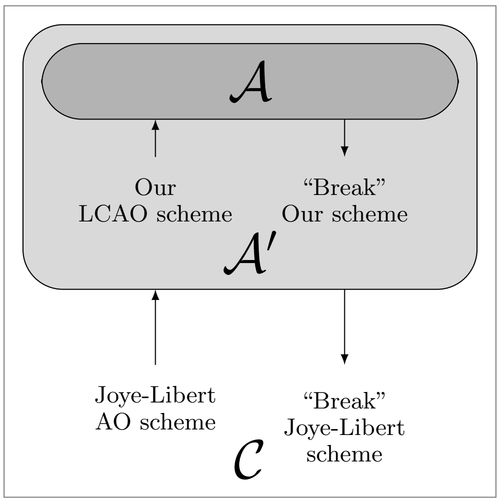

Fig. 3. High-level overview of our scheme's LCAO security proof.

**Queries** Handle queries from $\mathcal{A}$:

***Weight Query*** When $\mathcal{A}$ submits a weight query $t$, choose weights $\omega_j^{(t)}$, $1 \leq j \leq m$ and random values $r_j \in \mathbb{Z}_N$, $1 \leq j \leq m$, and return encryptions

$$(N+1)^{\omega_j^{(t)}} r_j^N \pmod{N^2},\ 1 \leq j \leq m$$

to $\mathcal{A}$.

***Combine Query*** When $\mathcal{A}$ submits combine query $(i, t, x_{i,1}^{(t)}, \ldots, x_{i,m}^{(t)})$, choose weights $\omega_j^{(t)}, 1 \leq j \leq m$ if not already chosen for time $t$, and make an AO encryption query $(i, t, \sum_{j=1}^m x_{i,j}^{(t)} \omega_j^{(t)})$ to $\mathcal{C}$. The received response is of the form $(N+1)^{\sum_{j=1}^m x_{i,j}^{(t)} \omega_j^{(t)}} H(t)^{sk_i}$; multiply it by $r^N$ for a random $r \in \mathbb{Z}_N$ and return

$$(N+1)^{\sum_{j=1}^m x_{i,j}^{(t)} \omega_j^{(t)}} r^N H(t)^{sk_i} \pmod{N^2}$$

to $\mathcal{A}$.

***Compromise Query*** When $\mathcal{A}$ submits compromise query $i$, make the same compromise query $i$ to $\mathcal{C}$, and return the recieved secret key $sk_i$ to $\mathcal{A}$.

**Challenge** When $\mathcal{A}$ submits challenge series

$$\langle (i, t^*, x_{i,1}^{(t^*)(0)}, \ldots, x_{i,m}^{(t^*)(0)}) \mid i \in S \rangle$$

and

$$\langle (i, t^*, x_{i,1}^{(t^*)(1)}, \ldots, x_{i,m}^{(t^*)(1)}) \mid i \in S \rangle ,$$

choose weights $\omega_j^{(t^*)}, 1 \leq j \leq m$ for time $t^*$ and submit AO challenge series

$$\langle (i, t^*, \sum_{j=1}^m x_{i,j}^{(t^*)(0)} \omega_j^{(t^*)}) \mid i \in S \rangle$$

and

$$\langle (i, t^*, \sum_{j=1}^m x_{i,j}^{(t^*)(1)} \omega_j^{(t^*)}) \mid i \in S \rangle ,$$

to $\mathcal{C}$. The received response if of the form

$$\langle (N+1)^{\sum_{j=1}^m x_{i,j}^{(t^*)(b)} \omega_j^{(t^*)}} H(t^*)^{sk_i} \mid i \in U \rangle ,$$

for an unknown $b \in \{0, 1\}$. Multiply series elements by $r_i^N$, $1 \leq i \leq n$ for randomly chosen $r_i \in \mathbb{Z}_N$ and return

$$\langle (N+1)^{\sum_{j=1}^m x_{i,j}^{(t^*)(b)} \omega_j^{(t^*)}} r_i^N H(t^*)^{sk_i} \mid i \in U \rangle$$

to $\mathcal{A}$.

**Guess** When $\mathcal{A}$ makes guess $b'$, make the same guess $b'$ to $\mathcal{C}$.

In the above construction, $\mathcal{C}$ follows the Joye-Libert scheme from Section 4.3 exactly, and to $\mathcal{A}$, $\mathcal{A}'$ behaves identically to our scheme described in Section 5.2. Since $\mathcal{A}'$ runs in polynomial-time to security parameter when $\mathcal{A}$ does, and no non-neglibile advantage adversary to $\mathcal{C}$ exists [9], we conclude that no non-negligible advantage adversary $\mathcal{A}$ exists. That is, there exists a negligible function $\eta$, such that

$$\mathsf{Adv}^{LCAO}(\mathcal{A}) \leq \eta(\kappa)$$

for security parameter $\kappa$. □

## 6 Private Localisation Preliminaries

The localisation filter we introduce requires real-valued inputs and functions, and relies on a non-linear measurement model. We make use of a real-number encoding scheme which supports the required homomorphic operations and an algebraic reformulation of the Extended Kalman Filter which reduces the filter update step to use only these operations.

### *6.1 Integer Encoding for Real Numbers*

In the encryption scheme introduced, weights and values are restricted to integers and all operations are computed $\pmod N$, thus bounding meaningful inputs to $\{x : x \in \mathbb{Z}_N\}$. For this reason, a quantisation and integer mapping method for real numbers is required for their encryption and homomorphic processing. We quantise with a generalised Q number format [12] due to implementation simplicity and applicability.

We define a subset of rational numbers in terms of a range $r \in \mathbb{N}$ and fractional precision $f \in \mathbb{N}$. This contrasts with the common definition given in terms of total bits $a$ and fractional bits $b$ [12,14,8], but allows for a direct mapping to integer ranges which are not powers of two. Rational subset $\mathbb{Q}_{r,f}$ is given by

$$\mathbb{Q}_{r,f} = \left\{ q : f^{-1} | q \wedge - \left\lfloor \frac{r}{2} \right\rfloor \leq q < \left\lfloor \frac{r}{2} \right\rfloor \right\} ,$$

and we quantise any real number $x$ by taking the nearest rational $q \in \mathbb{Q}_{r,f}$. That is, $\arg\min_{q \in \mathbb{Q}_{r,f}} |x - q|$. In this form, mapping rationals $\mathbb{Q}_{r,f}$ to the encryption scheme range $\mathbb{Z}_N$ is achieved by choosing $r = N$, and handling negatives with modulo arithmetic. In addition, we note that the Q number format requires a precision factor $f$ to be removed after each encoded multiplication, which is not supported by our encryption scheme. This is captured by a third parameter $m$; the number of *additional* multiplication factors to add or remove from encodings.

The combined quantisation and encoding function $\mathsf{E}_{r,f,m}(x)$ of a given a real number $x \in \mathbb{R}$, integer range $\mathbb{Z}_N$, and the desired scaling for $m$ prior encoded multiplications, is given by

$$\mathsf{E}_{N,f,m}(x) = \lfloor f^{m+1} x \rceil \pmod{N} . \tag{17}$$

Decoding of an integer $e \in \mathbb{Z}_N$, is given by

$$\mathsf{E}^{-1}_{N,f,m}(e) = \begin{cases} \dfrac{e \pmod{N}}{f^{m+1}}, & e \pmod{N} \leq \left\lfloor \dfrac{N}{2} \right\rfloor \\ -\dfrac{N - e \pmod{N}}{f^{m+1}}, & \text{otherwise} \end{cases} . \tag{18}$$

This encoding scheme provides the following homomorphic operations,

$$\mathsf{E}_{N,f,m}(a_1) + \mathsf{E}_{N,f,m}(a_2) \pmod{N} = \mathsf{E}_{N,f,m}(a_1 + a_2) \tag{19}$$

and

$$\mathsf{E}_{N,f,m}(a_1)\mathsf{E}_{N,f,m}(a_2) \pmod{N} = \mathsf{E}_{N,f,m+1}(a_1 a_2) , \tag{20}$$

noting that the modulus corresponds with the encrypted homomorphic operations in (15).

The choice of a high fractional precision $f$ may reduce quantisation errors introduced in (17), however, risks an overflow following too many multiplications. Given the largest number of expected multiplications $m_{max}$, and the largest expected decoded value $x$, the parameter should be chosen such that the following condition holds

$$\left| f^{m_{max}+1} x \right| < \left\lfloor \frac{N}{2} \right\rfloor .$$

In practice, $N$ is typically very large ($N > 2^{1024}$) and this condition can be ignored.

### *6.2 Extended Information Filter*

The Extended Information Filter (EIF) [11] is a reformulation of the Extended Kalman Filter (EKF), such that the update equations are simplified to sums of measurement information from each sensor. This requires the conversion of state estimate $\underline{\boldsymbol{x}}_k$ and estimate covariance $\mathbf{P}_k$ to the information vector and matrix

$$\underline{\boldsymbol{y}}_k = \mathbf{P}_k^{-1} \underline{\boldsymbol{x}}_k \quad \text{and} \quad \mathbf{Y}_k = \mathbf{P}_k^{-1} , \tag{21}$$

respectively. In this form, the update equations for $n$ measurements at time $k$, given process and measurement models (2) and (3), are given by

$$\underline{\boldsymbol{y}}_k = \underline{\boldsymbol{y}}_k + \sum_{i=1}^{n} \mathbf{H}_{k,i}^\top r_i^{-1} (z_{k,i} - h_i(\underline{\boldsymbol{x}}_k) + \mathbf{H}_{k,i} \underline{\boldsymbol{x}}_k) \tag{22}$$

and

$$\mathbf{Y}_k = \mathbf{Y}_k + \sum_{i=1}^{n} \mathbf{H}_{k,i}^\top r_i^{-1} \mathbf{H}_{k,i} , \tag{23}$$

with Jacobian

$$\mathbf{H}_{k,i} = \left. \frac{\partial h_i}{\partial \underline{x}} \right|_{\underline{\boldsymbol{x}}_k} \tag{24}$$

for sensors $1 \leq i \leq n$. The filter's prediction step can be computed by converting information vector $\underline{\boldsymbol{y}}_k$ and matrix $\mathbf{Y}_k$ back to state estimate and covariance

$$\underline{\boldsymbol{x}}_k = \mathbf{Y}_k^{-1} \underline{\boldsymbol{y}}_k \quad \text{and} \quad \mathbf{P}_k = \mathbf{Y}_k^{-1} , \tag{25}$$

before using the normal EKF prediction equations

$$\underline{\boldsymbol{x}}_k = \underline{f}(\underline{\boldsymbol{x}}_{k-1}) \tag{26}$$

and

$$\mathbf{P}_k = \mathbf{F}_k \mathbf{P}_{k-1} \mathbf{F}_k^\top , \tag{27}$$

with Jacobian

$$\mathbf{F}_k = \left. \frac{\partial \underline{f}}{\partial \underline{x}} \right|_{\underline{\boldsymbol{x}}_{k-1}} . \tag{28}$$

## 7 Private Localisation with Privacy-Preserving Sensors

Given our definitions for the aggregation scheme in Section 5.2 and preliminaries in Section 6, we can now construct a localisation method meeting our problem formulation in Section 3. We consider a navigator in the presence of $n$ range sensors, running a local filter with update steps computed by equations (22) and (23). Received information from each sensor $i$ consists of the measurement vector

$$\underline{\boldsymbol{i}}_{k,i} = \mathbf{H}_{k,i}^\top r_i^{-1} (\boldsymbol{z}_{k,i} - h_i(\underline{\boldsymbol{x}}_k) + \mathbf{H}_{k,i} \underline{\boldsymbol{x}}_k) \tag{29}$$

and the measurement matrix

$$\mathbf{I}_{k,i} = \mathbf{H}_{k,i}^\top r_i^{-1} \mathbf{H}_{k,i} \,, \tag{30}$$

at each time step $k$. In this form, all sensitive sensor information is exclusively found in $\underline{\boldsymbol{i}}_{k,i}$ and $\mathbf{I}_{k,i}$. Namely, their measurements $\boldsymbol{z}_{k,i}$, measurement variances $r_i$ and locations $\underline{s}_i$; captured in measurement functions $h_i$ and Jacobians $\mathbf{H}_{k,i}$.

Normal privacy-preserving aggregation encryption of $\underline{\boldsymbol{i}}_{k,i}$ and $\mathbf{I}_{k,i}$ at the sensors would require the navigator to disclose some location information in $\underline{\boldsymbol{x}}_k$, for the computing of Jacobians $\mathbf{H}_{k,i}$. Instead, we want to formulate $\underline{\boldsymbol{i}}_{k,i}$ and $\mathbf{I}_{k,i}$ such that the sensors can use an LCAO secure encryption scheme, with location information in $\underline{\boldsymbol{x}}_k$ as weights from the navigator. As was shown in Section 5.2, this guarantees that sensors cannot learn navigator location information $\underline{\boldsymbol{x}}_k$, and the navigator cannot learn individual sensor values $\underline{\boldsymbol{i}}_{k,i}$ and $\mathbf{I}_{k,i}$. From (29) and (30) we see that such a reformulation sets requirements on the possible measurement functions $h_i$.

## *7.1 Requirements for the Measurement Model*

From the definition of matrix multiplication, we know that matrices whose elements are linear combinations of weights can be multiplied to produce elements of new linear combinations, with weights dependant only on those from the original matrices. For example, consider dimension $N \times N$ matrices $\mathbf{A}$ and $\mathbf{B}$, where element $(i, j)$ in matrix $\mathbf{A}$ is of the form $\sum_{l=0}^{M} w_{l,i,j} a_{l,i,j}$ and similarly for matrix $\mathbf{B}$, form $\sum_{l=0}^{M} v_{l,i,j} b_{l,i,j}$. If we consider $w_{l,i,j}$ and $v_{l,i,j}$ as the weight terms, we can write element $(i, j)$ of a product matrix $\mathbf{AB}$, with weights dependant only on these, as

$$\begin{aligned}
&\sum_{l=0}^{N} \left( \left( \sum_{m=0}^{M} w_{m,i,j} a_{m,i,l} \right) \left( \sum_{m=0}^{M} v_{m,i,j} b_{m,l,j} \right) \right) \\
&\quad = \sum_{l=0}^{N} \left( \sum_{m=0}^{M} \sum_{n=0}^{M} w_{m,i,l} a_{m,i,l} v_{n,l,j} b_{n,l,j} \right) \\
&\quad = \sum_{l=0}^{N} \sum_{m=0}^{M} \sum_{n=0}^{M} (w_{m,i,l} v_{n,l,j}) a_{m,i,l} b_{n,l,j} \,.
\end{aligned} \tag{31}$$

Similarly, scalar multiplication also holds this property. It is therefore sufficient for $h_i$ and $\mathbf{H}_{k,i}$ to be linear combinations of information private to the navigator, for $\underline{\boldsymbol{i}}_{k,i}$ and $\mathbf{I}_{k,i}$ to be as well. Next, we will show how this can be achieved for the specific case of two-dimensional range-only localisation.

## *7.2 Localisation Measurement Model*

Recalling the state definition (1) and two-dimensional measurement model (3), we can see that $h_i$ *cannot* be rearranged to a linear combination of elements private to the navigator. In addition, the Jacobian of $h_i$,

$$\mathbf{H}_{k,i} = \begin{bmatrix} \frac{\boldsymbol{x}_k - s_{x,i}}{\sqrt{(\boldsymbol{x}_k - s_{x,i})^2 + (\boldsymbol{y}_k - s_{y,i})^2}} \\ \frac{\boldsymbol{y}_k - s_{y,i}}{\sqrt{(\boldsymbol{x}_k - s_{x,i})^2 + (\boldsymbol{y}_k - s_{y,i})^2}} \end{bmatrix} \,, \tag{32}$$

cannot be either. Instead, we consider the modified measurement functions

$$h_i'(\underline{\boldsymbol{x}}_k) = h_i(\underline{\boldsymbol{x}}_k)^2 \,. \tag{33}$$

A measurement function in this form allows rearrangement to a linear combination of navigator information, and thus of the corresponding modified measurement vector $\underline{\boldsymbol{i}}'_{k,i}$ and measurement matrix $\mathbf{I}'_{k,i}$ as well. The modified measurement functions $h_i'$ can be written as

$$\begin{aligned}
h_i'(\underline{\boldsymbol{x}}_k) &= \left\| \begin{bmatrix} \boldsymbol{x}_k & \boldsymbol{y}_k \end{bmatrix}^\top - \underline{s}_i \right\|^2 \\
&= (\boldsymbol{x}_k - s_{x,i})^2 + (\boldsymbol{y}_k - s_{y,i})^2 \\
&= \boldsymbol{x}_k^2 + \boldsymbol{y}_k^2 - 2 s_{x,i} \boldsymbol{x}_k - 2 s_{y,i} \boldsymbol{y}_k + s_{x,i}^2 + s_{y,i}^2
\end{aligned} \,, \tag{34}$$

and the corresponding Jacobians $\mathbf{H}'_{k,i}$ as

$$\mathbf{H}'_{k,i} = \begin{bmatrix} 2\boldsymbol{x}_k - 2s_{x,i} \\ 2\boldsymbol{y}_k - 2s_{y,i} \end{bmatrix} \,. \tag{35}$$

The above have been rearranged as linear combinations of the weights $\boldsymbol{x}_k^2$, $\boldsymbol{y}_k^2$, $\boldsymbol{x}_k$ and $\boldsymbol{y}_k$, which capture all the required information that is private to the navigator.

To show how $\underline{\boldsymbol{i}}'_{k,i}$ and $\mathbf{I}'_{k,i}$ can be formulated in a similar manner using (31), we require the existance of a modified measurement model of the form

$$\boldsymbol{z}'_{k,i} = h_i'(\underline{\boldsymbol{x}}_k) + \boldsymbol{v}'_{k,i} \,, \tag{36}$$

where $\boldsymbol{z}'_{k,i}$ is the modified measurement, and noise $\boldsymbol{v}'_{k,i}$ is white and Gaussian. The approximation of $\boldsymbol{z}'_{k,i}$ and $\boldsymbol{r}_{k,i}$ from original measurements and variances $\boldsymbol{z}_{k,i}$ and $r_i$ will be shown in Section 7.3.

With the existence of (36), measurement vector and measurement matrix linear combinations can be given by

$$\begin{aligned}\underline{\boldsymbol{i}}'_{k,i} &= \mathbf{H}'^{\top}_{k,i}\boldsymbol{r}^{-1}_{k,i}(\boldsymbol{z}'_{k,i} - h'_i(\underline{\boldsymbol{x}}_k) + \mathbf{H}'_{k,i}\underline{\boldsymbol{x}}_k) \\ &= \begin{bmatrix} (2\boldsymbol{r}^{-1}_{k,i})\boldsymbol{x}^3_k + (2\boldsymbol{r}^{-1}_{k,i})\boldsymbol{x}_k\boldsymbol{y}^2_k + (-\boldsymbol{r}^{-1}_{k,i}s_{x,i})\boldsymbol{x}^2_k \\ \quad + (-2\boldsymbol{r}^{-1}_{k,i}s_{x,i})\boldsymbol{y}^2_k + (2\boldsymbol{r}^{-1}_{k,i}\boldsymbol{z}'_{k,i})\boldsymbol{x}_k \\ \quad + (-2\boldsymbol{r}^{-1}_{k,i}s^2_{x,i})\boldsymbol{x}_k + (-2\boldsymbol{r}^{-1}_{k,i}s^2_{y,i})\boldsymbol{x}_k \\ \quad + (2\boldsymbol{r}^{-1}_{k,i}s^3_{x,i}) + (2\boldsymbol{r}^{-1}_{k,i}s_{x,i}s^2_{y,i}) \\ \quad + (-2\boldsymbol{r}^{-1}_{k,i}s_{x,i}\boldsymbol{z}'_{k,i}) \\ (2\boldsymbol{r}^{-1}_{k,i})\boldsymbol{y}^3_k + (2\boldsymbol{r}^{-1}_{k,i})\boldsymbol{x}^2_k\boldsymbol{y}_k + (-2\boldsymbol{r}^{-1}_{k,i}s_{y,i})\boldsymbol{x}^2_k \\ \quad + (-2\boldsymbol{r}^{-1}_{k,i}s_{y,i})\boldsymbol{y}^2_k + (2\boldsymbol{r}^{-1}_{k,i}\boldsymbol{z}'_{k,i})\boldsymbol{y}_k \\ \quad + (-2\boldsymbol{r}^{-1}_{k,i}s^2_{x,i})\boldsymbol{y}_k + (-2\boldsymbol{r}^{-1}_{k,i}s^2_{y,i})\boldsymbol{y}_k \\ \quad + (2\boldsymbol{r}^{-1}_{k,i}s_{y,i}s^2_{x,i}) + (2\boldsymbol{r}^{-1}_{k,i}s^3_{y,i}) \\ \quad + (-2\boldsymbol{r}^{-1}_{k,i}s_{y,i}\boldsymbol{z}'_{k,i}) \end{bmatrix}\end{aligned} \tag{37}$$

and

$$\begin{aligned}\mathbf{I}'_{k,i} &= \mathbf{H}'^{\top}_{k,i}\boldsymbol{r}^{-1}_{k,i}\mathbf{H}'_{k,i} \\ &= \begin{bmatrix}\alpha_{11} & \alpha_{12} \\ \alpha_{21} & \alpha_{22}\end{bmatrix},\end{aligned} \tag{38}$$

with

$$\begin{aligned}\alpha_{11} &= (4\boldsymbol{r}^{-1}_{k,i})\boldsymbol{x}^2_k + (-8\boldsymbol{r}^{-1}_{k,i}s_{x,i})\boldsymbol{x}_k + (4\boldsymbol{r}^{-1}_{k,i}s^2_{x,i}) \\ \alpha_{12} &= (4\boldsymbol{r}^{-1}_{k,i})\boldsymbol{x}_k\boldsymbol{y}_k + (-4\boldsymbol{r}^{-1}_{k,i}s_{y,i})\boldsymbol{x}_k + (-4\boldsymbol{r}^{-1}_{k,i}s_{x,i})\boldsymbol{y}_k \\ &\quad + (4\boldsymbol{r}^{-1}_{k,i}s_{x,i}s_{y,i}) \\ \alpha_{21} &= \alpha_{12} \\ \alpha_{22} &= (4\boldsymbol{r}^{-1}_{k,i})\boldsymbol{y}^2_k + (-8\boldsymbol{r}^{-1}_{k,i}s_{y,i})\boldsymbol{y}_k + (4\boldsymbol{r}^{-1}_{k,i}s^2_{y,i}).\end{aligned}$$

The above rearrangements result in $\underline{\boldsymbol{i}}'_{k,i}$ and $\mathbf{I}'_{k,i}$ as linear comabinations of the weights $\boldsymbol{x}^3_k$, $\boldsymbol{y}^3_k$, $\boldsymbol{x}^2_k\boldsymbol{y}_k$, $\boldsymbol{x}_k\boldsymbol{y}^2_k$, $\boldsymbol{x}^2_k$, $\boldsymbol{y}^2_k$, $\boldsymbol{x}_k\boldsymbol{y}_k$, $\boldsymbol{x}_k$ and $\boldsymbol{y}_k$, which again capture all the required information private to the navigator.

The final step required for the application of our LCAO-secure encryption scheme to the linear combinations (37) and (38) is the handling of the scheme's instance-stamp variable $t$, in (15). As six linear-combination aggregations occur at each time step $k$ (four element in $\mathbf{I}'_{k,i}$ and two in $\underline{\boldsymbol{i}}'_{k,i}$), it is required for the security of the scheme that $t$ be unique for each aggregation at each time. This is handled by setting $t$ to the concatenation

$$t = k \parallel i \parallel j \parallel b \tag{39}$$

for aggregation in row $i$ and column $j$, with $b = 0$ for aggregations in $\underline{\boldsymbol{i}}'_{k,i}$ and $b = 1$ otherwise.

**Remark 6** *The solutions* (37) *and* (38) *have been derived for two-dimensional localisation, but can be similarly extended to the three-dimensional case. We note however, the additional cost with increasing dimension. The number of weights is increased in the rearranged functions $h'_i$, and therefore combinatorially increased in $\underline{\boldsymbol{i}}'_{k,i}$ and $\mathbf{I}'_{k,i}$ by* (31). *This results in a combinatorial increase in weights, and therefore communication, with respect to the number of state parameters required for computing functions $h_i$.*

## *7.3 Range Measurement Modification*

In Section 7.2, we assumed the existance of modified sensor measurements $\boldsymbol{z}'_{k,i}$ with Gaussian noise and variances $\boldsymbol{r}_{k,i}$ in the modified measurement model (36). In practice, conversion of range measurements $\boldsymbol{z}_{k,i}$ to $\boldsymbol{z}'_{k,i}$ is complicated by the noise term $\boldsymbol{v}_{k,i} \sim \mathcal{N}(0, r_i)$ in (3). Squaring the range measurement produces

$$\begin{aligned}\boldsymbol{z}^2_{k,i} &= (h_i(\underline{\boldsymbol{x}}_k) + \boldsymbol{v}_{k,i})^2 \\ &= h_i(\underline{\boldsymbol{x}}_k)^2 + 2h_i(\underline{\boldsymbol{x}}_k)\boldsymbol{v}_{k,i} + \boldsymbol{v}^2_{k,i} \\ &= h'_i(\underline{\boldsymbol{x}}_k) + 2h_i(\underline{\boldsymbol{x}}_k)\boldsymbol{v}_{k,i} + \boldsymbol{v}^2_{k,i},\end{aligned} \tag{40}$$

with new noise term, $2h_i(\underline{\boldsymbol{x}}_k)\boldsymbol{v}_{k,i} + \boldsymbol{v}^2_{k,i}$, now dependant on the measurement function $h_i$ and no longer white or Gaussian. We can compute the mean of the new noise term (a function of white Gaussian term $\boldsymbol{v}_{k,i}$) as

$$\mathrm{E}\{2h_i(\underline{\boldsymbol{x}}_k)\boldsymbol{v}_{k,i} + \boldsymbol{v}^2_{k,i}\} = r_i \tag{41}$$

and define modified measurements as

$$\begin{aligned}\boldsymbol{z}'_{k,i} &= \boldsymbol{z}^2_{k,i} - r_i \\ &= h_i(\underline{\boldsymbol{x}}_k)^2 + 2h_i(\underline{\boldsymbol{x}}_k)\boldsymbol{v}_{k,i} + \boldsymbol{v}^2_{k,i} - r_i \\ &= h'_i(\underline{\boldsymbol{x}}_k) + \boldsymbol{v}'_{k,i},\end{aligned} \tag{42}$$

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

$$\mathrm{Var}\{\boldsymbol{v}'_{k,i}\} = 4h_i(\underline{\boldsymbol{x}}_k)^2 r_i + 2r^2_i \tag{43}$$

and is dependant on $h_i(\underline{\boldsymbol{x}}_k)$. We also wish to approximate this noise using values available to the sensor, $\boldsymbol{z}_{k,i}$ and $r_i$, while providing a conservative estimate of the true variance so as not to affect the convergence of the EIF. While replacing $h_i(\underline{\boldsymbol{x}}_k)$ with $\boldsymbol{z}_{k,i}$ only provides a conservative estimate in the case

$$\begin{aligned}\boldsymbol{z}_{k,i} &> h_i(\underline{\boldsymbol{x}}_k) \\ \boldsymbol{z}_{k,i} - h_i(\underline{\boldsymbol{x}}_k) &> 0 \\ \boldsymbol{v}_{k,i} &> 0,\end{aligned} \tag{44}$$

which cannot be guaranteed, we can instead provide a conservative estimate with 95% confidence, by shifting measurement $\boldsymbol{z}_{k,i}$ by two standard deviations $\sqrt{r_i}$. Thus, the modified measurement variance for sensor $i$ at time $k$ is conservatively approximated by

$$\boldsymbol{r}_{k,i} = 4(\boldsymbol{z}_{k,i} + 2\sqrt{r_i})^2 r_i + 2r_i^2 \,. \tag{45}$$

In addition, we note that the distribution of noise in (42) approaches a Gaussian when true ranges are sufficiently larger than measurement variances, as shown by

$$\begin{aligned} & r_i << h_i(\underline{\boldsymbol{x}}_k) \\ \Longrightarrow\ & 2h_i(\underline{\boldsymbol{x}}_k)\boldsymbol{v}_{k,i} + \boldsymbol{v}_{k,i}^2 - r_i \approx 2h_i(\underline{\boldsymbol{x}}_k)\boldsymbol{v}_{k,i} \\ \Longrightarrow\ & \boldsymbol{v}'_{k,i} \approx 2h_i(\underline{\boldsymbol{x}}_k)\boldsymbol{v}_{k,i} \,, \end{aligned} \tag{46}$$

and conclude with the following assumption.

**Assumption 7** *Given property* (46)*, and our* 95% *confindence in the conservativness of variance estimates* (45)*, we assume that range measurements $\boldsymbol{z}_i$ are in general large enough to produce sufficiently Gaussian modified measurement noise $\boldsymbol{v}'_{k,i}$ and its variance estimated sufficiently conservative to be used in the EIF and measurement model* (36)*.*

## *7.4 Algorithm*

In this section, we piece together our LCAO secure encryption scheme in Section 5.2 with the modified measurement model in Sections 7.2 and 7.3 and define our privacy-preserving localisation algorithm.

The privacy-preserving localisation filter can be described by the following steps.

**Setup** The $\mathsf{Setup}$ algorithm from Section 5.2 is executed by a trusted third party, $N$ and $H$ are made known to the navigator and all sensors, and the navigator and sensor secret keys, $sk_0 = \lambda$ and $sk_i$, $1 \le i \le n$, are distributed accordingly. Additionally, we assume that fractional precision $f$ from Section 6.1 is also a public parameter and made known to the navigator and sensors. We will thus simplify the encoding notation $\mathsf{E}_{N,f,m}(\cdot)$ to $\mathsf{E}_m(\cdot)$.

**Prediction** The navigator computes the typical EKF prediction equations before encrypting state variables and broadcasting to sensors. This is given by Algorithm 1.

**Measurement** Sensors modify their measurements before homomorphically computing measurement vectors and matrices $\underline{\boldsymbol{i}}'_{k,1}$ and $\mathbf{I}'_{k,1}$, encrypting them for aggregation, and sending them back to the navigator. Care must be taken when encoding during homomorphic operations to ensure fractional factor $m$ satisfies (19). This has been described by Algorithm 2.

**Update** The navigator aggregates measurement vectors and matrices before decrypting them. The typical EIF update equations can then be computed. This is shown in Algorithm 3.

**Algorithm 1** Navigator Prediction

```
 1: procedure PREDICTION($\underline{\boldsymbol{x}}_{k-1}$, $\mathbf{P}_{k-1}$, $\underline{f}$, $\mathbf{Q}$, $N$)
 2:     Compute $\mathbf{F}_k$ by (28)
 3:     $\underline{\boldsymbol{x}}_k \leftarrow \underline{f}(\underline{\boldsymbol{x}}_{k-1})$
 4:     $\mathbf{P}_k \leftarrow \mathbf{F}_k \mathbf{P}_{k-1} \mathbf{F}_k^\top + \mathbf{Q}$
 5:     for $\boldsymbol{\omega}$ in $\langle \boldsymbol{x}_k^3, \boldsymbol{y}_k^3, \boldsymbol{x}_k^2\boldsymbol{y}_k, \boldsymbol{x}_k\boldsymbol{y}_k^2, \boldsymbol{x}_k^2, \boldsymbol{y}_k^2, \boldsymbol{x}_k\boldsymbol{y}_k, \boldsymbol{x}_k, \boldsymbol{y}_k \rangle$ do
 6:         Compute $\mathsf{E}_0(\boldsymbol{\omega})$ by (17)
 7:         Compute $\mathcal{E}_{pk_0}(\mathsf{E}_0(\boldsymbol{\omega}))$ by (14)
 8:         Broadcast $\mathcal{E}_{pk_0}(\mathsf{E}_0(\boldsymbol{\omega}))$ to sensors
 9:     end for
10:     return $\underline{\boldsymbol{x}}_k, \mathbf{P}_k$
11: end procedure
```

**Algorithm 2** Measurement at Sensor $i$

```
 1: procedure MEASUREMENT($i$, $s_{x,i}$, $s_{y,i}$, $r_i$, $N$, $H$)
 2:     Measure $\boldsymbol{z}_{k,i}$
 3:     Compute $\boldsymbol{z}'_{k,i}$ by (42)
 4:     Compute $\boldsymbol{r}_{k,i}$ by (45)
 5:     while Navigator broadcasting do
 6:         Recieve $\mathcal{E}_{pk_0}(\mathsf{E}_0(\boldsymbol{x}_k^3))$
 7:         Recieve remaining weights in the form above
 8:     end while
 9:     Let $\beta_{vw}^{(i)}$ represent an encryption of element $(v, w)$ in $\underline{\boldsymbol{i}}'_{k,i}$ from (37)
10:     $\beta_{11}^{(i)} \leftarrow \mathcal{E}_{pk_0}(\mathsf{E}_0(\boldsymbol{x}_k^3))^{\mathsf{E}_0(2\boldsymbol{r}_{k,i}^{-1})}$
          $\mathcal{E}_{pk_0}(\mathsf{E}_0(\boldsymbol{x}_k\boldsymbol{y}_k^2))^{\mathsf{E}_0(2\boldsymbol{r}_{k,i}^{-1})} \mathcal{E}_{pk_0}(\mathsf{E}_0(\boldsymbol{x}_k^2))^{\mathsf{E}_0(-\boldsymbol{r}_{k,i}^{-1} s_{x,i})}$
          $\mathcal{E}_{pk_0}(\mathsf{E}_0(\boldsymbol{y}_k^2))^{\mathsf{E}_0(-2\boldsymbol{r}_{k,i}^{-1} s_{x,i})} \mathcal{E}_{pk_0}(\mathsf{E}_0(\boldsymbol{x}_k))^{\mathsf{E}_0(2\boldsymbol{r}_{k,i}^{-1} \boldsymbol{z}'_{k,i})}$
          $\mathcal{E}_{pk_0}(\mathsf{E}_0(\boldsymbol{x}_k))^{\mathsf{E}_0(-2\boldsymbol{r}_{k,i}^{-1} s_{x,i}^2)} \mathcal{E}_{pk_0}(\mathsf{E}_0(\boldsymbol{x}_k))^{\mathsf{E}_0(-2\boldsymbol{r}_{k,i}^{-1} s_{y,i}^2)}$
          $(N+1)^{\mathsf{E}_1(2\boldsymbol{r}_{k,i}^{-1} s_{x,i}^3)} (N+1)^{\mathsf{E}_1(2\boldsymbol{r}_{k,i}^{-1} s_{x,i} s_{y,i}^2)}$
          $(N+1)^{\mathsf{E}_1(-2\boldsymbol{r}_{k,i}^{-1} s_{x,i} \boldsymbol{z}'_{k,i})} H(k \| 1 \| 1 \| 0) \pmod{N^2}$
11:     Compute $\beta_{21}^{(i)}$ using (37) and (15) in the form above
12:     for $v \leftarrow 1$ to $2$ do
13:         Send $\beta_{v1}$ to the navigator
14:     end for
15:     Let $\alpha_{vw}^{(i)}$ represent an encryption of element $(v, w)$ in $\mathbf{I}'_{k,i}$ from (38)
16:     $\alpha_{11}^{(i)} \leftarrow \mathcal{E}_{pk_0}(\mathsf{E}_0(\boldsymbol{x}_k^2))^{\mathsf{E}_0(4\boldsymbol{r}_{k,i}^{-1})}$
          $\mathcal{E}_{pk_0}(\mathsf{E}_0(\boldsymbol{x}_k))^{\mathsf{E}_0(-8\boldsymbol{r}_{k,i}^{-1} s_{x,i})} (N+1)^{\mathsf{E}_1(4\boldsymbol{r}_{k,i}^{-1} s_{x,i}^2)}$
          $H(k \| 1 \| 1 \| 1) \pmod{N^2}$
17:     Compute remaining $\alpha_{vw}^{(i)}$ using (38) and (15) in the form above
18:     for $v \leftarrow 1$ to $2$ do
19:         for $w \leftarrow 1$ to $2$ do
20:             Send $\alpha_{vw}^{(i)}$ to the navigator
21:         end for
22:     end for
23: end procedure
```

**Algorithm 3** Navigator Update

```
1: procedure UPDATE($\underline{\boldsymbol{x}}_k$, $\mathbf{P}_k$, $N$, $\lambda$)
2:     for $v \leftarrow 1$ to $2$ do
3:         Receive $\beta_{v1}^{(i)}$ from each sensor $1 \le i \le n$
4:     end for
5:     for $v \leftarrow 1$ to $2$ do
6:         for $w \leftarrow 1$ to $2$ do
7:             Receive $\alpha_{vw}^{(i)}$ from each sensor $1 \le i \le n$
8:         end for
9:     end for
10:    Let $\beta_{vw}$ represent an encryption of element $(v, w)$ in $\sum_{i=1}^{n} \underline{\boldsymbol{i}}'_{k,i}$
11:    for $v \leftarrow 1$ to $2$ do
12:        $\beta_{v1} \leftarrow \prod_{i=1}^{n} \beta_{v1}^{(i)}$
13:        Compute $\mathcal{D}_{sk_0}(\beta_{v1})$ with $\lambda$ by (16)
14:        Compute $\mathsf{E}_1^{-1}(\mathcal{D}_{sk_0}(\beta_{v1}))$ by (18)
15:    end for
16:    Construct $\sum_{i=1}^{n} \underline{\boldsymbol{i}}'_{k,i}$ from decoded decryptions above
17:    Let $\alpha_{vw}$ represent an encryption of element $(v, w)$ in $\sum_{i=1}^{n} \mathbf{I}'_{k,i}$
18:    for $v \leftarrow 1$ to $2$ do
19:        for $w \leftarrow 1$ to $2$ do
20:            $\alpha_{vw} \leftarrow \prod_{i=1}^{n} \alpha_{vw}^{(i)}$
21:            Compute $\mathcal{D}_{sk_0}(\alpha_{vw})$ with $\lambda$ by (16)
22:            Compute $\mathsf{E}_1^{-1}(\mathcal{D}_{sk_0}(\alpha_{vw}))$ by (18)
23:        end for
24:    end for
25:    Construct $\sum_{i=1}^{n} \mathbf{I}'_{k,i}$ from decoded decryptions above
26:    $\underline{\boldsymbol{y}}_k \leftarrow \mathbf{P}_k^{-1}\underline{\boldsymbol{x}}_k + \sum_{i=1}^{n} \underline{\boldsymbol{i}}'_{k,i}$
27:    $\mathbf{Y}_k \leftarrow \mathbf{P}_k^{-1} + \sum_{i=1}^{n} \mathbf{I}'_{k,i}$
28:    $\underline{\boldsymbol{x}}_k \leftarrow \mathbf{Y}_k^{-1}\underline{\boldsymbol{y}}_k$
29:    $\mathbf{P}_k \leftarrow \mathbf{Y}_k^{-1}$
30:    return $\underline{\boldsymbol{x}}_k, \mathbf{P}_k$
31: end procedure
```

Algorithms 1, 2 and 3 have been summarised graphically in Figure 4, where encryptions and encodings of vectors and matrices have been denoted with $\mathcal{E}_{pk_0,sk_i}(\cdot)$ and $\mathsf{E}_m(\cdot)$ for brevity, and represent element-wise operations with the same parameters.

### *7.5 Leakage*

With the algorithm defined, we can analyse the localisation leakage when the strictly defined LCAO leakage is taken into account. We recall the assumptions on participant capabilities made in Section 3.2. Consistent broadcasting is in line with the LCAO definition and required to guarantee the scheme's leakage. Honest-but-curious sensors are not required for scheme security, but avoid malicious sensors gaining prior knowledge about remaining sensor measurements, which may be achieved by detecting changes in real measurements after reporting false measurements to the navigator. As stated in Section 3.2, misbehaving sensors are a known problem

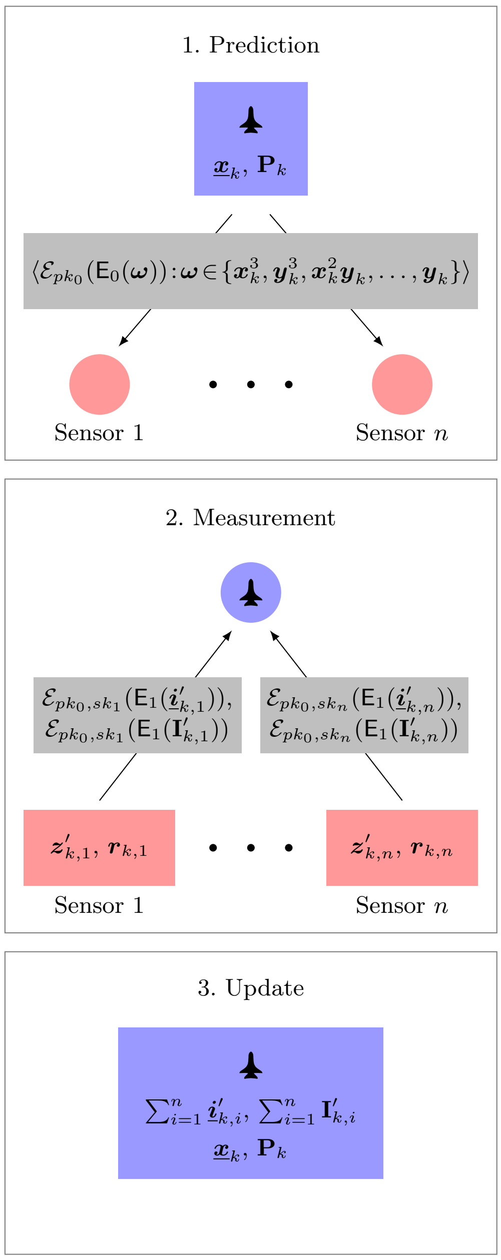
1. Prediction
$\underline{\boldsymbol{x}}_k$, $\mathbf{P}_k$
$\langle \mathcal{E}_{pk_0}(\mathsf{E}_0(\boldsymbol{\omega})) : \boldsymbol{\omega} \in \{\boldsymbol{x}_k^3, \boldsymbol{y}_k^3, \boldsymbol{x}_k^2\boldsymbol{y}_k, \dots, \boldsymbol{y}_k\}\rangle$
Sensor 1
Sensor $n$
2. Measurement
$\mathcal{E}_{pk_0,sk_1}(\mathsf{E}_1(\underline{\boldsymbol{i}}'_{k,1}))$, $\mathcal{E}_{pk_0,sk_1}(\mathsf{E}_1(\mathbf{I}'_{k,1}))$
$\mathcal{E}_{pk_0,sk_n}(\mathsf{E}_1(\underline{\boldsymbol{i}}'_{k,n}))$, $\mathcal{E}_{pk_0,sk_n}(\mathsf{E}_1(\mathbf{I}'_{k,n}))$
$\boldsymbol{z}'_{k,1}$, $\boldsymbol{r}_{k,1}$
$\boldsymbol{z}'_{k,n}$, $\boldsymbol{r}_{k,n}$
Sensor 1
Sensor $n$
3. Update
$\sum_{i=1}^{n} \underline{\boldsymbol{i}}'_{k,i}$, $\sum_{i=1}^{n} \mathbf{I}'_{k,i}$
$\underline{\boldsymbol{x}}_k$, $\mathbf{P}_k$

Fig. 4. The summary of steps involved in our proposed privacy-preserving EIF.

which complicates homomorphic operations and active malicious sensors are not considered in the scope of this work.

The aggregation operation of our encryption scheme leaks the sums of sensor measurement vectors and matrices $\sum_{i=1}^{n} \underline{\boldsymbol{i}}'_{k,i}$ and $\sum_{i=1}^{n} \mathbf{I}'_{k,i}$. However, as stated in Remark 4, weights chosen by a corrupted navigator mean that individual sums of coefficients can be leaked as well. That is, sums of coefficients in (37) and

(38) (*i.e.* $\sum_{i=1}^{n} 2\boldsymbol{r}_{k,i}^{-1}$, $\sum_{i=1}^{n} -\boldsymbol{r}_{k,i}^{-1}s_{x,i}$, $\sum_{i=1}^{n} -2\boldsymbol{r}_{k,i}^{-1}s_{x,i}$, $\sum_{i=1}^{n} 2\boldsymbol{r}_{k,i}^{-1}\boldsymbol{z}'_{k,i}, \dots$) are leaked to a corrupted navigator.

From the leaked coefficient sums, we see that all information private to the sensors, namely their measurements $\boldsymbol{z}'_{k,i}$ and locations $\underline{s}_i$, are present only in sums

$$\sum_{i=1}^{n} \boldsymbol{z}'_{k,i}\,, \; \sum_{i=1}^{n} s_{x,i} \text{ and } \sum_{i=1}^{n} s_{y,i}\,. \tag{47}$$

Therefore, we conclude that the navigator can *at most* learn the sums of sensor private data, which in practice corresponds to the average sensor measurements and locations.

**Remark 8** *In the context of our localisation scheme, leakage of the incorporated LCAO secure encryption scheme corresponds to the leakage of average sensor measurements and locations, given by the sums in* (47).

## 8 Simulation and Results

To demonstrate our proposed approach, we have implemented the algorithm in Section 7.4 and simulated the measurement and navigation of an object following a linear, time-invariant, process model in two-dimensional space.

Code was written in the C programming language, using the MPI library [4] to support simultaneous computations by sensors and navigator as different processes. The OpenSSL library's [5] mask generation function MGF1 and hash function SHA256 were used to implement the required hash function $H$, while the Libpaillier library [1] was used for the implementation of the Paillier encryption scheme. Additionally, GNU libraries GSL [2] and GMP [3] were used for algebraic operations and the handling of multiple-precision integers, respectively. Simulation execution was performed on a 3.00GHz Intel i7-9700 CPU, and run on the Windows Subsystem for Linux (WSL), where no real-time kernel was ensured but sufficiently consistent runtimes were observed for the purposes of our simulation.

Recalling Assumption 7, modified measurement noise resembles a Gaussian distribution when measurements are sufficiently large. We considered 4 layouts with varying sensor distances when running our simulations to observe differences in estimation error when measurements are of different sizes. The considered layouts have been plotted and named in Figure 5. To evaluate the accuracy of our method, we have run 100 simulations, all with 50 filter iterations, for each of the sensors layouts with unmodified sensor measurement variances $r_i = 5$ metres. Due to the large encoding modulus $N$, no notable advantage is gained when choosing a low fractional

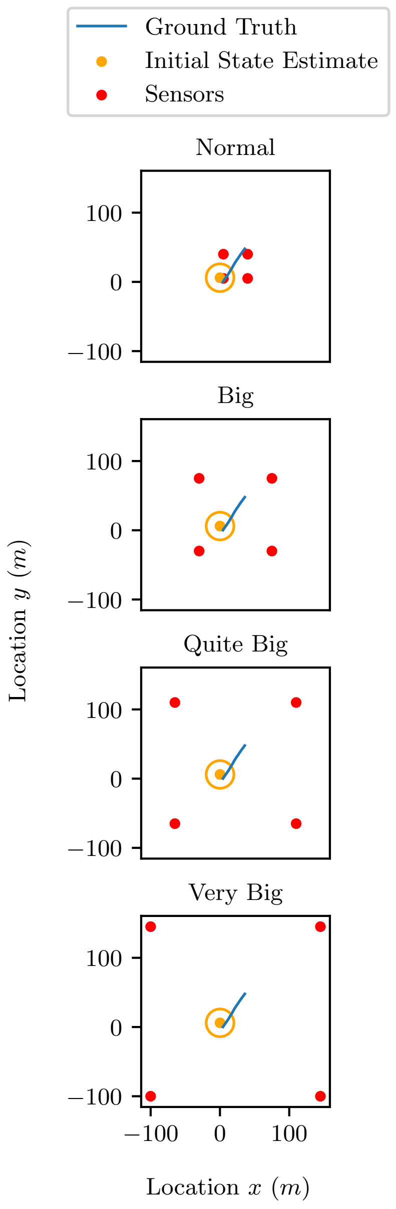

Fig. 5. Layouts Normal Big

precision factor $f$, thus, $f = 2^{32}$ was chosen to ensure minimal loss of precision when compared to the floating-point computations in the unencrypted EIF. The plotted average RMSE of our proposed method as well as that of the unencrypted EIF filter can be seen in Figure 6. From the results, it can be seen that sensor distance has little effect on the performance of the filter and that the measurement modification in Section 7.3 has almost no effect on the performance when compared to a normal EIF filter. The robustness of the privacy-preserving method is likely due to the rationality of Assumption 7, the relative size of measurements to the sensor variances in our simulations, and the sufficiently large fractional

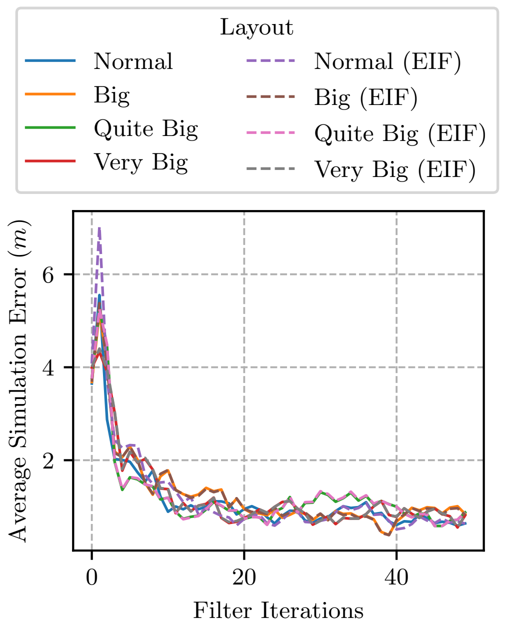

Fig. 6. Layout errors

precision.

In addition to accuracy, computational requirements are an important factor in the adoption of novel cryptographic methods. Figure 7 shows the average simulation runtimes when encryption key bit lengths (*i.e.* bit lengths of $N$) and the number of sensors are varied. Every plotted time is again the average of 100 simulations, each running for 50 filter iterations. As sensor and navi-

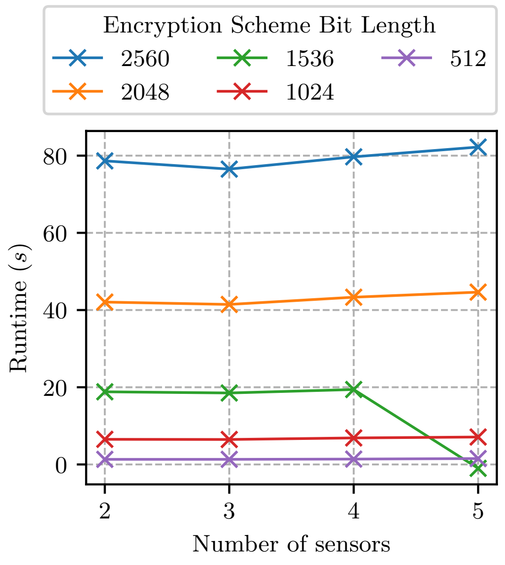

Fig. 7. Timing

gator computations run in parallel, increases in runtime from additional sensors are primarily due to the longer aggregation operation at the navigator. The predominant computational cost is dependant on the encryption scheme key size, which must be chosen such that sufficient security is provided by the scheme. The current recommendation for a secure implementation relying on prime factorisation (difficulty of factorising $N$) is to use 2048 bit-long keys [6].

## 9 Conclusion

To develop a model-based localisation algorithm which preserves navigator and sensor privacy, we first defined the novel notion of an LCAO secure encryption scheme and gave a provably secure implementation based on the Joye-Libert aggregation scheme. This scheme was then used in our proposed private EIF, where sensors compute measurement information homomorphically such that it can be privately used by a navigator. Our privacy-preserving estimation method may find uses in a variety of untrusted distributed localisation environments including airspaces and autonomous vehicle networks as well as those where alternative measurement models satisfying our defined linearity requirements can be applied. Possible future work in this topic includes the expanding of the LCAO security notion to *ensure* that the same weights are broadcast to all sensors, the relaxing of the Gaussian modified noise assumption and the exploring of implications behind an active sensor attacker model.

## Acknowledgements

Partially supported by ZIM

## References

[1] Libpaillier, 2010.

[2] GSL - GNU Scientific Library, 2019.

[3] GMP - The GNU Multiple Precision Arithmetic Library, 2020.

[4] Open MPI, 2020.

[5] OpenSSL, 2020.

[6] Elaine Barker, Lily Chen, Allen Roginsky, Apostol Vassilev, Richard Davis, and Scott Simon. Recommendation for pair-wise
key establishment using integer factorization cryptography. Technical Report NIST SP 800-56Br2, National Institute of Standards and Technology, Gaithersburg, MD, March 2019.

[7] M. Chase, H. Chen, J. Ding, S. Goldwasser, S. Gorbunov, J. Hoffstein, K. Lauter, S. Lokam, D. Moody, T. Morrison, and A. Sahai. Security of Homomorphic Encryption. *Technical Report, HomomorphicEncryption.org, Redmond WA, USA*, 2017.

[8] Farhad Farokhi, Iman Shames, and Nathan Batterham. Secure and Private Control Using Semi-Homomorphic Encryption. *Control Engineering Practice*, 67:13–20, 2017.

[9] Marc Joye and Benoît Libert. A Scalable Scheme for Privacy-Preserving Aggregation of Time-Series Data. In David Hutchison, Takeo Kanade, Josef Kittler, Jon M. Kleinberg, Friedemann Mattern, John C. Mitchell, Moni Naor, Oscar Nierstrasz, C. Pandu Rangan, Bernhard Steffen, Madhu Sudan, Demetri Terzopoulos, Doug Tygar, Moshe Y. Vardi, Gerhard Weikum, and Ahmad-Reza Sadeghi, editors, *Financial Cryptography and Data Security*, volume 7859, pages 111–125. Springer Berlin Heidelberg, Berlin, Heidelberg, 2013.

[10] J. Katz and Y. Lindell. *Introduction to Modern Cryptography: Principles and Protocols.* Chapman & Hall, 2008.

[11] Peter S. Maybeck. *Stochastic Models, Estimation, and Control.* Academic Press, August 1982.

[12] E. L. Oberstar. Fixed-Point Representation & Fractional Math. *Oberstar Consulting*, 9, 2007.

[13] P. Paillier. Public-Key Cryptosystems Based on Composite Degree Residuosity Classes. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques (EUROCRYPT)*, pages 223–238. Springer, 1999.

[14] Moritz Schulze Darup, Adrian Redder, and Daniel E. Quevedo. Encrypted Cooperative Control Based on Structured Feedback. *IEEE Control Systems Letters*, 3(1):37–42, January 2019.

[15] Elaine Shi, T-H Hubert Chan, and Eleanor Rieffel. Privacy-Preserving Aggregation of Time-Series Data. *Proc. NDSS.*, page 17, 2011.

## A Indistinguishability under Chosen Plaintext Attack (IND-CPA)

A public-key encryption scheme is defined by the tuple of algorithms $(\mathsf{Setup}, \mathsf{Enc}, \mathsf{Dec})$, defined as

$\mathsf{Setup}(\kappa)$ On input of security paramater $\kappa$, generate public key $pk$ and secret key $sk$

$\mathsf{Enc}(pk, x)$ Encryption of value $x$ is computable using the public key $pk$, obtaining $\mathcal{E}_{pk}(x)$.

$\mathsf{Dec}(sk, \mathcal{E}_{pk}(x))$ Decryption of value $x$ is computable using the secret key $sk$.

The security game between attacker and challenger for IND-CPA is given by

**Setup** The challenger runs the $\mathsf{Setup}$ algorithm and gives public key $pk$ to the attacker

**Encryptions** The attacker may compute encryptions using the public key $pk$.

**Challenge** Next, the attacker chooses two values

$$x^{(0)} \text{ and } x^{(1)}$$

and gives them to the challenger. The challenger then chooses a random bit $b \in \{1, 0\}$ and returns the encryption

$$\mathcal{E}_{pk}(x^{(b)}).$$

**More Encryptions** The attacker can now compute more encryptions with the public key $pk$.

**Guess** At the end, the attacker outputs a bit $b'$ and wins the game only if $b' = b$. The advantage of an attacker $\mathcal{A}$ is defined as

$$\mathsf{Adv}^{IND-CPA}(\mathcal{A}) := \left| \mathrm{P}[b' = b] - \frac{1}{2} \right| .$$

## B Aggregator Obliviousness (AO)

An aggregator oblivious encryption scheme is defined by the tuple of algorithms $(\mathsf{Setup}, \mathsf{Enc}, \mathsf{AggDec})$, defined as

$\mathsf{Setup}(\kappa)$ On input of security paramater $\kappa$, generate public parameters $\mathsf{pub}$, the user private keys $sk_i$, $1 \leq i \leq n$, and the aggregator's private key $sk_0 = -\sum_{i=1}^{n} sk_i$.

$\mathsf{Enc}(t, sk_i, x_i^{(t)})$ At time $t$, user $i$ computes and obtains the encrypted value $y_i^{(t)} = \mathcal{E}_{sk_i}(x_i^{(t)})$ using its secret key $sk_i$.

$\mathsf{AggDec}(t, sk_0, y_1^{(t)}, \ldots, y_n^{(t)})$ At time $t$, the aggregator computes the aggregation of values $\sum_{i=1}^{n} x_i^{(t)}$ using its private key $sk_0$.

The security game between attacker and challenger for AO is given by

**Setup** The challenger runs the $\mathsf{Setup}$ algorithm and gives public parameters $\mathsf{pub}$ to the attacker

**Queries** The attacker can now submit queries that are answered by the challenger. The types of queries are:

(1) *Combine Queries:* The attacker chooses a tuple $(i, t, x_i^{(t)}$ such that for any two chosen combine query tuples $(i, t, x_i^{(t)})$ and $(i', t', x_{i'}'^{(t')}$, the following condition holds:

$$i = i' \wedge t = t' \implies x_i^{(t)} = x_{i'}'^{(t')} .$$

They are given back the encryption of the value $\mathcal{E}_{sk_i}(x_i^{(t)})$ encrypted under the secret key $sk_i$.

(2) *Compromise queries:* The attacker chooses $i$ and receives the secret key $sk_i$. The aggregator's secret key may also be compromised (when choosing $i = 0$).

**Challenge** Next, the attacker chooses a time $t^*$, and a subset of users $S \subseteq U$ where $U$ is the complete set of users for which no combine queries, for time $t^*$, and no compromise queries, are made for the duration of the game. The attacker then chooses two series of tuples

$$\langle (i, t^*, x_i^{(t^*)(0)}) \,|\, i \in S \rangle$$

and

$$\langle (i, t^*, x_i^{(t^*)(1)}) \,|\, i \in S \rangle ,$$

and gives them to the challenger. In the case that $0 \in S$ (*i.e.* the aggregator is compromised) and $S = U$, it is additionally required that

$$\sum_{i \in S} x_i^{(t^*)(0)} = \sum_{i \in S} x_i^{(t^*)(1)} .$$

The challenger then chooses a random bit $b \in \{1, 0\}$ and returns encryptions

$$\langle \mathcal{E}_{sk_i}(x_i^{(t^*)(b)}) \mid i \in S \rangle .$$

**More Queries** The attacker can now submit more queries, so long as the queries do not break the requirements in the Challenge stage. That is, $S \subseteq U$.

**Guess** At the end, the attacker outputs a bit $b'$ and wins the game only if $b' = b$. The advantage of an attacker $\mathcal{A}$ is defined as

$$\mathsf{Adv}^{AO}(\mathcal{A}) := \left| \mathrm{P}[b' = b] - \frac{1}{2} \right| .$$