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

—

Introduce localisation, filtering and the need for privacy.

Examples of environments where privacy is relevant and concrete examples where lack of privacy could have large costs

Methods for introducing security and privacy include differential privacy methods and encryption methods.

Differential privacy involves using statistical noise as security to make individual users' information cannot be deduced. Often requires a trusted aggregator, although secure aggregation methods exist. always requires noising result such that the outcome is not exact (a problem in localisation).

Encryption schemes involve formal indistinguishability proofs typically over bits or integers. They rely on computationally hard problems involving security parameters of a sufficiently large size; therefore the additional computational requirements of using encryption schemes should be pointed out and what this means in a real-time distributed sensor system. Continuing, explain public-key cryptography applicability to distributed systems; difference to symmetric schemes. Homomorphic encryption power and use case. Why FHE isn't used often, why additive partially homomorphic encryption is.

Advancements in function providing encryption schemes such as homomorphic encryption have also led to several other types of schemes which have found uses in signal processing. Private aggregation schemes allow the secure computation of the sum of encrypted values originating from different parties, leaking only the final result. When considering such multi-party encryption protocols, for-

mal security definitions must now also incorporate the added dangers of colluding malicious parties, and lead to new notions of security. For example Aggregator Obliviousness (AO) is typically proven for private aggregation schemes, while alternatives such as Private Weighted Secure Aggregator Obliviousness (pWSAO) exist for other specific use-cases.

Another example of function providing encryption, and a generalisation of private aggregation, is called functional encryption (FE) and its distributed extension, multi-client functional encryption (MCFE), which allow the unencrypted result of an arbitrary function to be computed from encrypted inputs. General FE and MCFE are known to be quite computationally expensive (from meeting with ITI and student Johannes - need ref.) but alternatives providing only a subset of possibly computable function exist; for example, inner product encryption.

Several of the aforementioned encryption schemes have found uses in secure localisation, estimation, and control.

1.1   Relevant Literature on Encrypted Localisation and Estimation

Model-free localisation using homomorphic encryption examples include polygon thing, WSN examples which protect against adversaries but in the case of the WSN paper. don't preserve anchor privacy. Importantly, model-based filtering and localisation provide more accurate estimates and these are not applicable there.

Model-based estimation examples include Aristov paper (which requires a linear model, and a hierarchy of sensors), Farokhi paper (which requires the controller compute entirely in encrypted space and send input back to actuator - supporting only the cloud-as-a-service type architectures) and Alexandru paper (which implements a distributed control environment but requires a constant gain matrix K)

pWSAO achieved in Alexandru weighted aggregation, but requires redistributing keys at every timestep resulting in a costly operation, and a complicated communication protocol.

In addition to applying suitable encryption schemes to signal processing tasks, care must be taken when converting sensor output into an encryptable homomorphic format. As is the case with our proposed localisation method, real number sensor output does not trivially encode to integers such that the homomorphic properties provided by an additive encryption scheme over integers keep the underlying real numbers consistent. Methods for handling the encoding of real numbers such that they can be used in homomorphic encryption exist. Google

bignum adds power but risks overflow and leaks exponents, Farokhi leaks no information but allows only a single multiplication (extendable to more but each further multiplication limits the real number size and increases the risk of overflow).

Briefly describe navigator scenario and our contributions

–

In Section 2 we introduce the localisation problem and the restrictions on involved parties. Section 3 summarises cryptographic preliminaries and Sections 4 and 5 introduce a novel security notion of linear-combination aggregation obliviousness and an encryption scheme which satisfies it. Section 6 summarises localisation preliminaries, before we introduce our private localisation method in Section 7, using the introduced encryption scheme. In Section 8 we give and analyse simulation results, before concluding our work in Section 9.

1.2   Notation

Notation. Reminder to include symbols for: rounding, concatenation, "divides", vector norm, expected value and variance, ordered list

2   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

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

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

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

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 $\underline{x}_k$ and the navigator does not learn sensor locations $\underline{s}_i, 1 \leq i \leq n$ or their measurements 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.

2.1 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 1.

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 widespread 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.
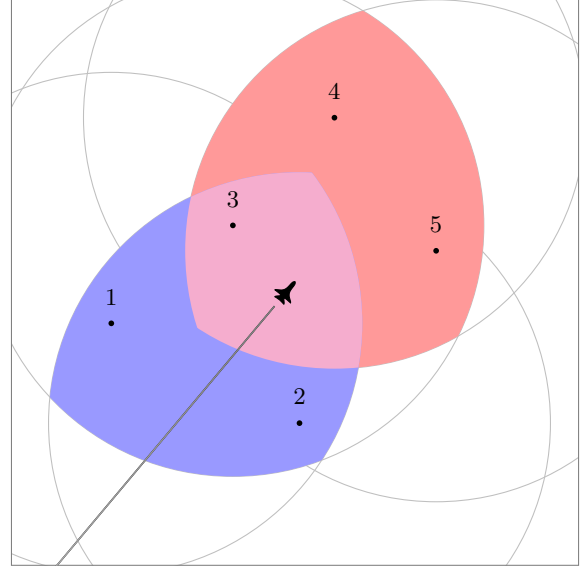

Fig. 1. An example of sensor subsets; $\{1, 2, 3\}$ and $\{3, 4, 5\}$, and the areas where the navigator is in range of all sensors within a subset.

2.2 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.

3 Cryptography Preliminaries

When defining our system 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.

3.1 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 [4]. The typical security notion for a homomorphic encryption scheme is Indistinguishability under Chosen Plaintext Attack (IND-CPA) [1].

Definition 1 *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) [9].

Definition 2 *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.

3.2 Paillier Encryption Scheme

The Paillier encryption scheme [7] 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 [4]. 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

$$c = g^m r^N \pmod{N^2} \qquad (6)$$

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

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

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. $\forall m_1, m_2 \in \mathbb{Z}_N$,

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

3.3 Joye-Libert Private Aggregation Scheme

The Joye-Libert private aggregation scheme [3] 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

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

removing all noise terms.

4 Private Linear-Combination Aggregation

For achieving the goal of private localisation defined in Section 2 we will require a multi-party protocol for computing linear combinations of weights, and their subsequent aggregation. In the context of navigation, m weights $\omega_j^{(t)}, 1 \leq j \leq m$ are first 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, at every time step t. This has been summarized in Figure 2. In Section 7 we will show how this protocol can be sufficient to compute measurement covariances and measurement vectors, and update the Extended Information Filter.

1. Broadcast and linear combinations step

Sensor 1

$(\omega_1^{(t)}, \ldots, \omega_m^{(t)})$

$y_1^{(t)} = \sum_{j=1}^{m} x_{1,j}^{(t)} \omega_j^{(t)}$

Navigator

$(\omega_1^{(t)}, \ldots, \omega_m^{(t)})$

$y_n^{(t)} = \sum_{j=1}^{m} x_{n,j}^{(t)} \omega_j^{(t)}$

Sensor n

2. Aggregation step

Sensor 1

$y_1^{(t)}$

Navigator

$\sum_{i=1}^{n} \sum_{j=1}^{m} x_{i,j}^{(t)} \omega_j^{(t)} = \sum_{i=1}^{n} y_i^{(t)}$
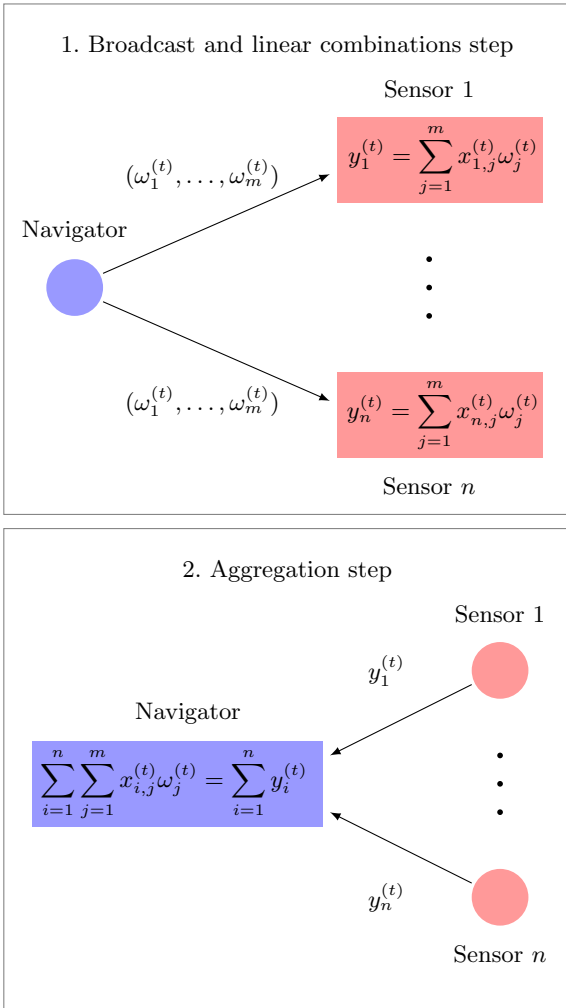
$y_n^{(t)}$

Sensor n

Fig. 2. Required linear-combination aggregation steps at time t

To perform the protocol from Figure 2 in a secure manner, we must first describe the security properties we aim to achieve. Broadly speaking, we want a cryptographic scheme which allows a time-series 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 timesteps, 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 timesteps. 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 3 *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.

4.1 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 κ, generate public parameters pub, number of weights m, the aggregator's public and private keys pk_0 and sk_0, and the remaining user private keys $sk_i, 1 \leq i \leq n$.

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

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.

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 Setup algorithm and gives 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 ω and computes an encryption of ω 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, $\mathcal{E}_{pk_0}(\omega_j^{(t)})$, $1 \le j \le m$.
(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 $(i', t', x_{i',1}'^{(t')}, \ldots, x_{i',m}'^{(t')})$, the following condition holds:

$$i = i' \wedge t = t' \implies x_{i,m}^{(t)} = x_{i',m}'^{(t')}, 1 \le j \le m.$$

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 $\omega_j^{(t^*)}$, $1 \le j \le m$ 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 4 *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* Setup, Enc *and* 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 Our Scheme

Our scheme is based on the Paillier and Joye-Libert schemes introduced in Section 3, 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.

Setup(κ) On input parameter κ, 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 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, $1 \le i \le n-1$ uniformly from $\mathbb{Z}_{N^2}$ and setting the last key as $sk_n = -\sum_{i=1}^{n-1} sk_i \pmod{N^2}$.

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

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

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

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

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

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

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

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

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

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

for some values $r_i, r_j \in \mathbb{Z}_N$, $1 \le j \le m$. 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.

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

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

The correctness of aggregation can be seen from

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

$$= H(t)^{\sum_{i=1}^n sk_i}.$$

$$\prod_{i=1}^n (N+1)^{\sum_{j=1}^m x_{i,j}^{(t)}\omega_j^{(t)}} r_i^N \pmod{N^2}$$

$$= (N+1)^{\sum_{i=1}^n \sum_{j=1}^m x_{i,j}^{(t)}\omega_j^{(t)}} r'^N \pmod{N^2},$$

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

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 $\pmod{N}$.

5.1 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 [7] 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 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 η_1 and η_2. (*i.e.* if we assume our scheme is not LCAO secure, then the Joye-Libert scheme is not AO secure.) This is a contradiction to the Joye-Libert AO proof in [3], and will thus conclude our proof. 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 4.1. 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 3.3.

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 $\mathcal{A}$.

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

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

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

to $\mathcal{A}$.

Combine Query When $\mathcal{A}$ submits combine query $(i, t, x_{i,1}^{(t)}, \ldots, x_{i,m}^{(t)})$, choose weights $\omega_j^{(t)}, 1 \le j \le m$ 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

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

and

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

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

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

and

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

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

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

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

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

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 3.3 exactly, and to $\mathcal{A}$, $\mathcal{A}'$ behaves identically to our scheme described in Section 5. Since $\mathcal{A}'$ runs in polynomial-time to security parameter when $\mathcal{A}$ does, and no non-neglibile advantage adversary to $\mathcal{C}$ exists [3], we conclude that no non-negligible advantage adversary $\mathcal{A}$ exists. That is, there exists a negligible function η, such that

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

for security parameter κ. $\quad\square$

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 [6] 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 [6,8,2], but allows for a direct mapping to integer ranges which are not powers of two. Rational subset $\mathbb{Q}_{r,f}$ is given by

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

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

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

$$\tag{18}$$

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) [5] 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 $\underline{x}_k$ and estimate covariance $\mathbf{P}_k$ to the information vector and matrix

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

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

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

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

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

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

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

before using the normal EKF prediction equations

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

and

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

with Jacobian

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

7 Private Localisation with Privacy-Preserving Sensors

Given our definitions for the aggregation scheme in Section 5 and preliminaries in Section 6, we can now construct a localisation method meeting our problem formulation in Section 2. 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

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

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 $\underline{i}_{k,i}$ and $\mathbf{I}_{k,i}$. Namely, their measurements $z_{k,i}$, 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 $\underline{i}_{k,i}$ and $\mathbf{I}_{k,i}$ at the sensors would require the navigator to disclose some location information in $\underline{x}_k$, for the computing of Jacobians $\mathbf{H}_{k,i}$. Instead, we want to formulate $\underline{i}_{k,i}$ and $\mathbf{I}_{k,i}$ such that the sensors can use an LCAO secure encryption scheme, with location information in $\underline{x}_k$ as weights from the navigator. As was shown in Section 5, this guarantees that sensors cannot learn navigator location information $\underline{x}_k$, and the navigator cannot learn individual sensor values $\underline{i}_{k,i}$ 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

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

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

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

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

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

and

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

with

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

The above rearrangements result in $\underline{\boldsymbol{i}}_{k,i}'$ and $\mathbf{I}_{k,i}'$ as linear comabinations of the weights $\boldsymbol{x}_k^3$, $\boldsymbol{y}_k^3$, $\boldsymbol{x}_k^2 \boldsymbol{y}_k$, $\boldsymbol{x}_k \boldsymbol{y}_k^2$, $\boldsymbol{x}_k^2$, $\boldsymbol{y}_k^2$, $\boldsymbol{x}_k$ 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 $\underline{i}'_{k,i}$), 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 $\underline{i}'_{k,i}$ and $b = 1$ otherwise.

Remark 5 *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 $\underline{i}'_{k,i}$ 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 $z'_{k,i}$ with Gaussian noise and variances $r_{k,i}$ in the modified measurement model (36). In practice, conversion of range measurements $z_{k,i}$ to $z'_{k,i}$ is complicated by the noise term $v_{k,i} \sim \mathcal{N}(0, r_i)$ in (3). Squaring the range measurement produces

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

with new noise term, $2h_i(\underline{x}_k)v_{k,i} + v^2_{k,i}$, 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 $v_{k,i}$) as

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

and define modified measurements as

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

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

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

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

$$
\begin{aligned}
z_{k,i} &> h_i(\underline{x}_k) \\
z_{k,i} - h_i(\underline{x}_k) &> 0 \\
v_{k,i} &> 0 ,
\end{aligned}
\tag{44}
$$

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

$$r_{k,i} = 4(z_{k,i} + 2\sqrt{r_i})^2 r_i + 2r^2_i . \tag{45}$$

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

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

and conclude with the following assumption.

Assumption 6 *Given property (46), and our 95% confidence in the conservativness of variance estimates (45), we assume that range measurements z_i are in general large enough to produce sufficiently Gaussian modified measurement noise $v'_{k,i}$ and its variance estimated sufficiently conservative to be used in the EIF and measurement model (36).*

7.4 Algorithm

Piece together the whole algorithm here. Give the algorithm as pseudocode (including encoding and encryption, possible as separate procedures)

7.5 Leakage

We recall the assumptions made in the problem formulation and our encryption scheme, and the allowed leakage of the encryption scheme. Sensors are honest but curious, meaning sensor values cannot be modified from what is measured, and the protocol is followed exactly, and that the navigator sends the same state information to all sensors at each time step. This is justified to an extent with wireless broadcasting being received by all sensors in a typical localisation setting.

Algorithm 1 Navigator Prediction Step

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** ω in $\langle \boldsymbol{x}_k^3, \boldsymbol{y}_k^3, \boldsymbol{x}_k^2 \boldsymbol{y}_k, \boldsymbol{x}_k \boldsymbol{y}_k^2, \boldsymbol{x}_k^2, \boldsymbol{y}_k^2, \boldsymbol{x}_k, \boldsymbol{y}_k \rangle$ **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 β_{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 Step

1: **procedure** UPDATE($\underline{\boldsymbol{x}}_k$, $\mathbf{P}_k$, N, λ)
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 β_{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 λ 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 α_{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 λ 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**

From the above restrictions, sensors learn only their measurements x_i, and the navigator learns only the sums $\sum_{i=1}^n H^\top R_i^{-1}(z_i - h(x) + Hx)$ and $\sum_{i=1}^n H^\top R_i^{-1} H$.

For malicious subsets not including the navigator, it is clear that they learn nothing more about the navigator location than what they can deduce given only their own measurements.

The navigator, which may maliciously choose weights at every time step, is always restricted to learning at most the sum $\sum_{i=1}^n a_i$ for some values a_i derived from sensor i's location and measurement. It is therefore intuitive to say that at most the navigator can learn the average of values kept private, that is, the navigator can at most learn the average position and location of sensors. The same can be said for any corrupted subset of honest but curious sensors and malicious navigator.

8 Results

Time results can be captured in one graph. Y-axis is time, X-axis is the number of sensors, each line (different colour) will show how the runtime changes as sensors are increased for different Paillier bit-sizes (at least 3: 512, 1024, 2048). Every data point should be the average over some X number of simulations.

Accuracy plots will describe error due to encoding and the average distance of the sensors to the navigator. All plots will use the same ground truth and initial state and covariance estimates (this way average error at each timestep from multiple runs makes sense):

Plot 1 will plot the RMSE of the average of X runs at each encoding size. A fixed layout of 4 mediumly spaced sensor will be used, and a fixed Paillier bit size.

Plot 2 will plot the RMSE as the average distance of sensors changes. Fixed encoding and Paillier bit size. Vary between 4 layouts where 4 sensors are either very close to the centre (and ground truth, and progressively further out)

Plot 3 will accompany plot 3 and display the 4 layouts (arrow for ground truth and points for the sensors).

Discuss the meaning of the small changes in accuracy, and runtimes when using the same encryption size, that is, point out it is a generally robust method.

9 Conclusion

Possible future work to consider: Hardware implementations, measurement handling which preservers Gaussian noise, or non-Gaussian noise methods, ways of sending less information form the navigator to the sensors at each time step, active sensor attacker model, different state encryptions received at sensors or a way to confirm they are the same.

References

[1] 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.

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

[3] 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.

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

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

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

[7] 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.

[8] 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.

[9] 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 κ, 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 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 (Setup, Enc, AggDec), defined as

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

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.

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 Setup algorithm and gives public parameters 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

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

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

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