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-

⋆ 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).

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

Section Summary

1.2 Notation

Notation. Reminder to include symbols for: rounding, concatenation, "divides", vector norm, vector indexing (also note starts at 0)

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 simplicty, but will derive methods suitable for an extension to the three-dimensional equivalent.

The navigator state is defined as

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

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

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

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

$$z_k = h_i(\hat{x}_k) + \boldsymbol{v}_k \,, \tag{3}$$

with noise $\boldsymbol{v}_k \sim \mathcal{N}(\underline{0}, \mathbf{R})$, while the measurement function h_i, for sensor i at location

$$\underline{s}_i = \begin{bmatrix} x_i & y_i \end{bmatrix}^\top , \tag{4}$$

is defined as

$$\begin{aligned} h_i(\hat{x}_k) &= \left\| \begin{bmatrix} \hat{x}_{k[0]} \\ \hat{x}_{k[2]} \end{bmatrix} - \underline{s}_i \right\| \\ &= \sqrt{(\hat{x}_{k[0]} - x_i)^2 + (\hat{x}_{k[2]} - y_i)^2} \,. \end{aligned} \tag{5}$$

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 $\hat{x}_k$ 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.

Consistent navigator broadcast Received information is always equal.

Honest-but-curious sensors We adopt the honest-but-curious attacker model for sensors in our localisation method. That is, sensors will be assumed to follow the derived procedure correctly, but may use any gained sensitive information for other purposes.

Computational capabilities The computational requirements of the navigator and sensors are predominantly defined by encryption, decryption and homomorphic operations. We make the implicit assumption that all involved parties are computationally capable of computing the described encryption scheme and filter.

A problem that arises in some navigation environments when constrained by the requirement that all sensors receive navigator broadcast information, is when some sensors are not within measurement range at all times. We note that this can be handled by considering subsets of sensors for which overlapping measurement ranges can be used as separate areas within which our localisation method can be performed using only the relevant subset of sensors. As a navigator moves from one such area to another, it will perform the protocol with the new set of sensors whose signal it is in range of.

2.2 Inherent Leakage

Leakage of information will be discussed in more detail in Section 7.5, where leakage formalised in the developed linear-combination aggregation scheme is considered in our localisation problem context. Accepted leakage of private information will be bound by the average of private sensor values. That is, the average location and average measurement of all honest sensors are at most what can be leaked to the navigator, while no information about the navigator state will be leaked to sensors.

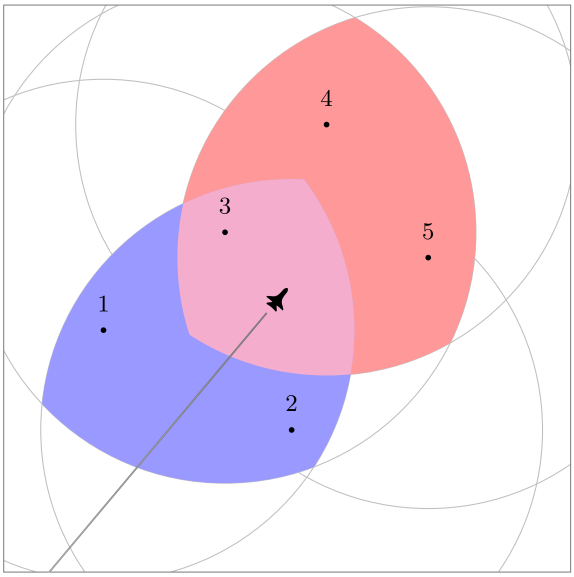

Fig. 1. An example of sensor subsets; $\{1, 2, 3\}$ and $\{3, 4, 5\}$, and the areas where the navigator is in range of all sensors within a subset.

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) [8].

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 [6] 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

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

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

$$\mathcal{D}(\mathcal{E}(m_1)\mathcal{E}(m_2) \pmod{N^2}) = m_1 + m_2 \pmod{N} \tag{8}$$

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

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

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, $1 \le i \le n$ 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

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

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

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

and

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

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.

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 $\omega_j^{(t)}, 1 \le j \le m$ 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 $1 \le i \le n$, 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.

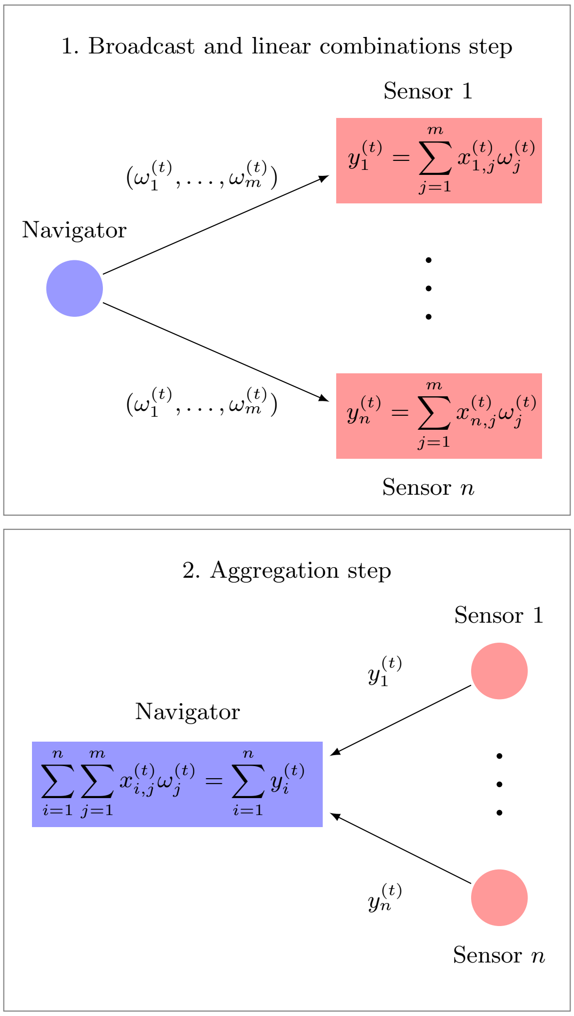

Fig. 2. Required linear-combination aggregation steps at time t

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 ω_j, $1 \le j \le m$ 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 $x_{i,j}^{(t)}$, $0 \le j \le m$, 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* $(1, 0, \dots, 0)$, *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, $1 \le i \le n$.

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

$\mathsf{CombEnc}(t, pk_0, sk_i, \mathcal{E}_{pk_0}(\omega_1^{(t)}), \dots, \mathcal{E}_{pk_0}(\omega_m^{(t)}), x_{i,1}^{(t)}, \dots, x_{i,m}^{(t)})$ 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 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 $(i', t', x_{i',1}^{\prime(t')}, \ldots, x_{i',m}^{\prime(t')})$, the following condition holds:

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

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

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

and

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

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

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

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

$$\langle \mathcal{E}_{pk_0,sk_i}(\sum_{j=1}^{m} x_{i,j}^{(t^*)(b)}\omega_j^{(t^*)}) \,|\, i \in S\rangle\,.$$

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* $\mathsf{CombEnc}$.

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 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.

$\mathsf{Setup}(\kappa)$ 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 $\mathsf{pub} = H$, 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, $1 \le i \le n-1$ uniformly from $\mathbb{Z}_{N^2}$ and setting the last key as $sk_0 = -\sum_{i=1}^{n} sk_i \pmod{N^2}$.

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

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

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

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

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

$\mathsf{AggDec}(t, pk_0, sk_0, y_1^{(t)}, \ldots, y_n^{(t)})$ 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

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

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

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 (mod 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 [6] 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

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

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 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

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

for security parameter κ. $\square$

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 [5] 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 [5,7,2], but allows for a direct mapping to integer ranges which are not powers of two. Rational subset $\mathbb{Q}_{r,f}$ is given by

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

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

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

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

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

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

EIF is a reformulation of the EKF where the update step is reduced to sums. Makes it especially easier for processing multiple measurements.

Prediction step can be computed in either the EIF form or the equivalent EKF state and covariance can be computed before performing the prediction step in EKF form and converting the result back.

7 Private Localisation with Privacy-Preserving Sensors

Using the EIF and distance sensors, we see that the required computation of measurement vector and matrices is reduced to computing $H^\top R^{-1}(z-h(x)+Hx)$ and $H^\top R^{-1}H$, for one-dimensional R and jacobian H, at each sensor. As this requires current state information x, relevant attributes need to be sent to the sensors.

We wish to encrypt current state information and use the encryption scheme defined to compute measurement information $H^\top R^{-1}(z - h(x) + Hx)$ and $H^\top R^{-1}H$ as a linear combination of said state information, and encrypt it for aggregation. This then allows the navigator to learn only the sums $\sum_{i=1}^{n} H^\top R_i^{-1}(z_i - h(x) + Hx)$ and $\sum_{i=1}^{n} H^\top R_i^{-1} H$, which is sufficient to update its position.

7.1 Requirements for Measurement Model

The requirement introduced above, is that measurement matrix and vector $H^\top R^{-1}(z-h(x)+Hx)$ and $H^\top R^{-1}H$ can be computed as linear combinations of functions involving only state vector elements. In particular, a linear combination of functions which depend only on information available to the navigator. In the measurement matrix and vector above, H is the measurement function Jacobian, and includes current state information as well.

As our encryption scheme only supports linear combinations of weights, the weights need to capture all non-linear computations involving state vector elements. From the definition of matrix multiplication, we see that any KxL matrix $D = ABC$, where B has dimensions MxN, has element $d_{i,j}$ defined by

$$d_{i,j} = \sum_{n=1}^{N} \sum_{m=1}^{M} a_{i,m} b_{m,n} c_{n,j} \,,$$

and can therefore be considered as a linear combination of all elements $a_{i,m}c_{n,j}$. Applying this to the measurement matrix and vector $H^\top R^{-1}(z - h(x) + Hx)$ and $H^\top R^{-1}H$, we require that all values of $h(x)$, and all elements in Jacobian H be linear combinations of functions dependant on state vector elements and nothing else.

We will show how this can be achieved for the specific case of 2-dimensional localisation, with range-only sensors.

7.2 Localisation Measurement Model

From our problem statement in Section 2, we specify the goal of private localisation in the 2-dimensional case, using range-only sensors. Given the current location estimate $x_{loc} = (x, y)$, which may comprise only part of the filter state estimate, and sensor location $s_{loc} = (s_x, s_y)$, the measurement equation is given by

$$\begin{aligned} h(x) &= \|x_{loc} - s_{loc}\| \\ &= \sqrt{(x - s_x)^2 + (y - s_y)^2} \,. \end{aligned}$$

The Jacobian H for the measurement function above at $x_{loc,0} = (x_0, y_0)$ is then given by

$$H = \begin{bmatrix} \frac{x_0 - s_x}{\sqrt{(x_0-s_x)^2+(y_0-s_y)^2}} \\ \frac{y_0 - s_y}{\sqrt{(x_0-s_x)^2+(y_0-s_y)^2}} \end{bmatrix} .$$

The problem with the above function h and Jacobian H is that they cannot be reformulated as a linear combination of weights, where weights do not depend on s_x and s_y.

Our proposed solution to this problem, is working with the adjusted measurement function $h'(x) = h(x)^2$. The method for modifying a normal range-sensor measurement in this way and handling the no-longer Gaussian measurement noise with variance dependant on distance, will be discussed in Section 7.3. While this transformation complicates the measurement model noise, it allows for the linear formulation required for the encryption scheme, and the privacy of $H'^\top R'^{-1}(z' - h(x) + Hx)$

and $H'^{\top}R'^{-1}H'$. The adjusted measurement function is now given by

$$\begin{aligned} h'(x) &= (\|x_{loc} - s_{loc}\|)^2 \\ &= (x - s_x)^2 + (y - s_y)^2 \\ &= x^2 + y^2 - 2s_x x - 2s_y y + s_x^2 + s_y^2 \end{aligned},$$

and Jacobian H' by

$$H' = \begin{bmatrix} 2x_0 - 2s_x \\ 2y_0 - 2s_y \end{bmatrix}.$$

The above function and jacobian elements are linear combinations of functions x, x^2, y, xy and y^2. From the previous section, we know we can now define $H'^{\top}R'^{-1}(z' - h(x) + Hx)$ and $H'^{\top}R'^{-1}H'$ in terms of more state vector dependant functions, in this case, x, x^2, x^3, y, xy, x^2y, y^2, xy^2 and y^3.

$$H'^{\top}R'^{-1}(z' - h(x) + Hx) = \begin{bmatrix} \begin{aligned}&(2R'^{-1}z')x + (-2R'^{-1}s_x z) + (-2R'^{-1}s_x^2)x \\ &\quad + (2R'^{-1}s_x^3) + (-2R'^{-1}s_y^2)x + (2R'^{-1}s_x s_y^2) \\ &\quad + (2R'^{-1})x^3 + (-R'^{-1}s_x)x^2 + (2R'^{-1})xy^2 \\ &\quad + (-2R'^{-1}s_x)y^2\end{aligned} \\ \begin{aligned}&(2R'^{-1}z)y + (-2R'^{-1}s_y z) + (-2R'^{-1}s_x^2)y \\ &\quad + (2R'^{-1}s_y s_x^2) + (-2R'^{-1}s_y^2)y + (2R'^{-1}s_y^3) \\ &\quad + (2R'^{-1})x^2y + (-2R'^{-1}s_y)x^2 + (2R'^{-1})y^3 \\ &\quad + (-2R'^{-1}s_y)y^2\end{aligned} \end{bmatrix}$$

and

$$H'^{\top}R'^{-1}H' = \begin{bmatrix} \alpha_{11} & \alpha_{12} \\ \alpha_{21} & \alpha_{22} \end{bmatrix},$$

with

$$\begin{aligned} \alpha_{11} &= (4R'^{-1})x^2 + (-8R'^{-1}s_x)x + (4R'^{-1}s_x^2) \\ \alpha_{12} &= (4R'^{-1})xy + (-4R'^{-1}s_y)x + (-4R'^{-1}s_x)y \\ &\quad + (4R'^{-1}s_x s_y) \\ \alpha_{21} &= \alpha_{12} \\ \alpha_{22} &= (4R'^{-1})y^2 + (-8R'^{-1}s_y)y + (4R'^{-1}s_y^2). \end{aligned}$$

The above formulation allows the application of our encryption scheme to the private aggregation of the measurement matrix and vector above. The scheme's encrypted weights x, x^2, x^3, y, xy, x^2y, y^2, xy^2 and y^3 are computed at each time step by the navigator, and each element of the measurement matrix and vector above is computed as a linear combination and prepared for aggregation by the sensors.

Lastly, since we aggregate multiple linear combinations at each time step k (each element in the result matrix and vector comes from a separate aggregation), we require unique hash values for each aggregation. This is handled by setting time variable t as concatenation $k\|i\|j$ for each element $x_{i,j}$. Aggregation of each element is now performed under a different encryption scheme "time" step, when computing (15).

The solution above has been derived for the two-dimensional localisation case but it is easy to see that a similar solution would work in three-dimensional space. The cost of increasing localisation dimension however, is the increased number of state vector element functions required for broadcast by the navigator, and length of linear combination computed by sensors.

7.3 Range Measurement Modification

The conversion of measurement function $h'(x) = h(x)^2$ is in practice complicated by measurement noise manipulation. As we only know $z = h(x) + n$, where $n\,\mathcal{N}(0, \sigma)$, we can estimate the new measurement as z^2. This results in the new measurement

$$\begin{aligned} z^2 &= (h(x) + n)^2 \\ &= h(x)^2 + 2nh(x) + n^2, \\ &= h'(x) + 2nh(x) + n^2 \end{aligned}$$

where noise term $2nh(x) + n^2$ has mean σ^2 and standard deviation $2(2h(x)^2\sigma^2 + \sigma^4)$. We account for the non-zero mean by offsetting the squared measurement and taking this as the adjusted measurement model

$$\begin{aligned} z' &= h'(x) + 2nh(x) + n^2 - \sigma^2 \\ &= h'(x) + n' \end{aligned},$$

with variance of n' approximated as $2(2z^2\sigma^2 + \sigma^4)$, by using the original measurement z as an unbiased estimate for $h(x)$.

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.

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.

Acknowledgements

Partially supported by zim

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

[6] 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.

[7] 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.

[8] 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 ($\mathsf{Setup}, \mathsf{Enc}, \mathsf{AggDec}$), defined as

$\mathsf{Setup}(\kappa)$ On input of security paramater κ, generate public parameters 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 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

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