OTTO VON GUERICKE
UNIVERSITÄT
MAGDEBURG

Otto von Guericke University
Institute for Intelligent Cooperating Systems
Autonomous Multisensor Systems (AMS)

Digital Engineering
Practical Project Report
September 5, 2021

NAME	**MATRICULATION**
Lakshmi Harika Nakka	230191
Sai Teja Thalluri	230192
Venkata Subba Raju Kanumuri	230195

Privacy-Preserving Localization and Privileged Sensor Fusion in a Camera Network

ams.ovgu.de

Privacy-Preserving Localization and Privileged Sensor Fusion in a Camera Network

– Digital Engineering: Practical Project Report–

September 5, 2021

Abstract

Distributed state estimation and localization methods have become increasingly prevalent in modern networked tracking systems, as traditional distributed sensor localization methods often involve the leakage of sensor or the state information during the estimation, thereby compromising the privacy of participants, henceforth breaching data privacy. This report centers on the implementation of a cryptographically secure localization scheme, to preserve the privacy of the participants. This involves designing a small-scale camera network, using Raspberry Pi nodes, which are set up in a distributed and synchronous network, to track a laser pointer (object). Privacy-Preserving formulation of the information filter is implemented to send the measurements securely between the nodes, aggregate the measurements and estimate the state of the object . Paillier encryption scheme, a cryptographic secure method, responsible for ensuring the privacy of all the input data, has been implemented. Measurements from the sensors are aggregated homomorphically, employing partially homomorphic addition operation at a local hub, thereby, restricting the central node holding the secret key, which does not know the individual sensor measurements thereby ensuring the privacy of the members.

Contents

1 Introduction

Privacy-Preserving Localization methods have engrossed researchers lately while working in distributed sensor environments [2].Information leakage rate, being a crucial metric, reflects the security in a wireless communication system. It is often a challenge to shield the sensor or state information from unintended recipients as they can be operated by different parties, or multiple external sensors may have to track a single object. The sensor data and parameters are typically gathered centrally, implying that the participants have to trust each other. Not only sensor data, but sensing modalities, sensor positions, and hardware details might be leaked to the aggregator (Central node). Privacy-preserving aggregation scheme combined with cryptographically secure localization schemes can possibly ensure preserving the privacy of the participants.

We localize an object (Red laser pointer) which ideally follows a linear constant velocity model. Initial state and positional coordinates of the laser pointer are obtained with respect to time using object detection algorithm. This laser pointer is localized using distributed image sensors (4 Raspberry Pi camera modules) rigidly fixed and establishing a fully connected distributed network. These measurements are captured at every sensor further sent to another sensor, either hierarchically or to a local hub which aggregates and encrypts the measurements before decrypting them at the central node.

We run then the state estimation filter which takes the aggregated values to estimate the state of the object. However, we don't want the leakage of individual sensor measurements to be known to any untrusted third party who can manipulate the data thereby giving us false estimates resulting in poor localization. Not only the third party, but individual sensors should not learn or have any prior knowledge of the other sensor's data in order to maintain privacy.

An encryption scheme is employed which should encrypt the data at each individual sensor. An aggregation operation/ scheme has to be implemented which meets the security notions thereby restricting the attackers to manipulate the data before estimating and localizing the laser. Additionally, an estimation encrypted version of a filter has to be developed which updates the measurements in the filter.

Problem statements which we would like to address with this report :

- How Privacy-preserving aggregation scheme (Paillier encryption scheme) can possibly ensure in preserving the privacy of the participants?
- How well we can implement an accurate localization filter (Privacy preserving Information filter), to leverage efficient multi-sensor state estimation, which heavily relies on the possibility to algebraically manipulate sensor data when nodes exchange data with each other?

The rest of the paper is organised as follows: In section 2, we discuss about Object detection, Networking (Distributed and Synchronization implementation), Kalman filter and information filter, Central processing, Privacy-Preserving cryptographic encryption scheme. Section 3 discusses our approach to acknowledge our problem statements in detail including the implementation. Section 5 states results from our experiment, followed by section 6 which concludes the report.

2 Background and Related Work

2.1 Object Tracking

As part of local processing the initial step is to detect and track an object, which in our case is a red laser light. The object is to be captured at the individual sensor separately and measurements produced will be later used as a suitable input for our filter. This can be done using various algorithms such as YOLO, SSD, Retina-Net, etc to detect the red laser light. An other effective way for object detection is **Color Model-HSV** which stands for Hue, Saturation and Value, is available in Computer Vision library Opencv(CV2). This model is often used in applications which run on Computer Vision. HSV tends to describe the features of the captured image better when compared to RGB model by finding the exact position of red spot in the image (Video capture).

An other advantage of using Opencv(CV2) is that we can ensure calibration of each camera with respect to world coordinate system and, use perspective transformation to align the 3-Dimensional video capture on a 2-Dimensional plane to produce a measurement in a form suitable for the filter.

2.2 Networking

Network allows communication between the Raspberry Pi nodes. Distribution and synchronisation implementation are accomplished as part of this section. This can be done using socket programming. In socket programming[4] we have Transmission Control Protocol (TCP), a connection oriented and User Datagram Protocol (UDP), a connection less network.

User Datagram Protocol (UDP) is a loss-tolerating and low-latency communication channel between two sockets which do not require a verified connection. Regardless of whether the data is being recieved by the recepient or not, it continuously sends datagarm.

Transmission Control Protocol(TCP) is a transport layer protocol, this interface is commonly known as Sockets interface, which is used between web clients and web servers for distributed and synchronised systems. Most internet applications use TCP as it provides reliable, full-duplex, and connection-oriented data transmission.

Below are the advantages of TCP over UDP:

- Data re-transmission : TCP segments are transferred from sender to receiver. Occasionally, few segments might fail to reach the destination. Unlike User Datagram Protocol(UDP), TCP sends acknowledgements to the user, which aids to retrieve the missing segments.

- Unique Identification: TCP communicates with each computer using a unique IP address, to distinguish them when they are set up in a common network.
- Detecting Errors: Error detection in the TCP is generally done through 3 steps, using the checksum, re-transmission and acknowledgement.
- In-order Delivery: The order in which packets of data being transferred might get lost in between. TCP takes necessary steps to rearrange them in the order sent.

We also need to check the effective processing of information at each time step, between our local server and the central server by implementing synchronous communication which is also possible in TCP communication.

2.3 Kalman & Information Filtering

2.3.1 Kalman Filter

The Kalman filter [8] produces an estimate of the state of the system as an average of the system's predicted state and of the new measurement. This result of the weighted average is a new state estimate that always lies between the predicted and measured state. This process is repeated at every time step, with the new estimate and its covariance informing the prediction used in the following iteration. This means that the Kalman filter works recursively and requires only the previous estimation, rather than the entire log of a system's state to estimate a new state.

The Kalman filter is a concrete inference scheme to estimate the state $x_k \in R_N$ of a dynamic linear system $x_{k+1} = A_k x_k^e + B_k u_k + w_k$ where $k \in N$ denotes the discrete time.

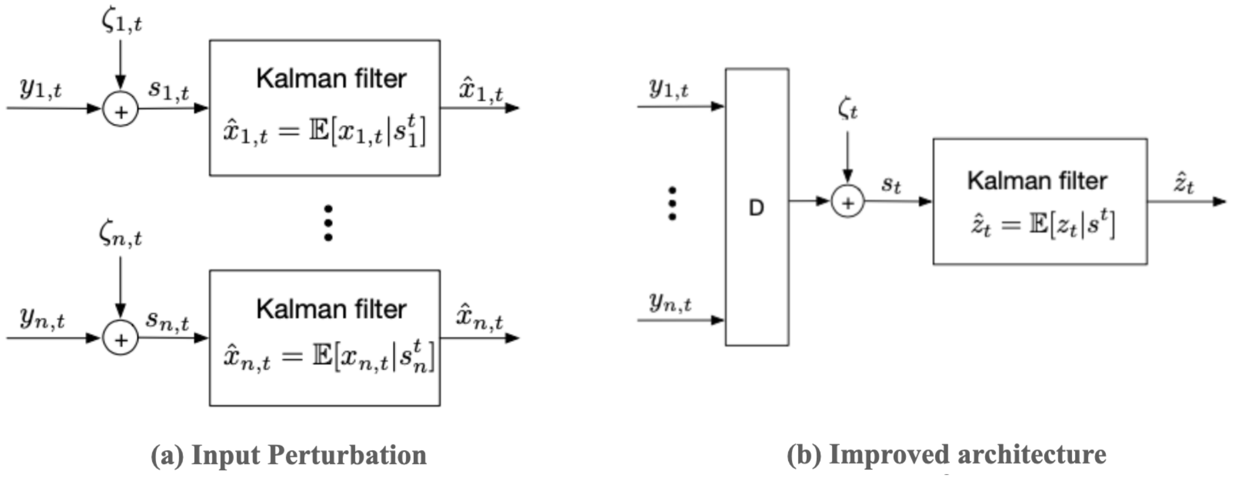


Figure 1: Differential private Kalman filtering architectures[3]

The process dynamics are characterized by the system matrix $A_k \in R_n * R_n$, Uncertainties affecting the state transition are given by the zero-mean with noise $w_k \sim N(0, Q)$ with covariance matrix $P_k \in R_n * R_n$. Observations of the state are provided by a network of $N \in N$ sensor nodes. An estimate x^e_k is computed with the aid of the measurements $z_k \in Rm$. The corresponding model $z_k = H_k.x_k + v_k$ is assumed to be linear with observation matrix $H_k \in R_m{*}_n$ and noise $v_k \sim N(0, R)$ with co variance $R_k \in R_m * R_m$. The Kalman filter is initialized with a prior estimate x^e_0 and prior covariance matrix P_{e0} and consists of prediction and update steps. In the previous step, the process model is used to compute a predicted estimate.

$$x^P_{k+1} = A_k x^e_k + B_k u_k$$

with error co variance matrix

$$P^P_{k+1} = A_k P^e_k A^T_k + Q_k$$

The update of the predicted estimate x is given by

$$x^e_{k+1} = x^P_{k+1} + K_{k+1} * y_{k+1}$$

With the updated error covariance matrix

$$P^e_{k+1} = (1 - K_{k+1} H_{k+1}) P^p_{k+1}$$

Where the kalman gain

$$K_{k+1} = P^P_{k+1} H^T_{k+1} S^{-1}_{k+1}$$

In distributed sensor networks, it is difficult to update the above equations and estimate the state by inserting multiple measurements is not suitable and becomes complicated.

Rather, these estimated sate and the estimated error covariance equations can simply be updated by converting the measurements and the error covariance matrices in the form of measurement vectors and the measurement matrices.

$$I_k = H^T_K R^{-1}_k H_k$$

$$i_k = H^T_K R^{-1}_k z_k$$

These measurement vectors and matrices from multiple sensors can be summed up and updated with the information matrix and the information vectors in the update step of the information filter.

$$Y_{k/k} = Y_{k/k-1} + \Sigma^N_{j=1} I_{K,j}$$

$$y_{k/k} = y_{k/k-1} + \Sigma^N_{j=1} i_{K,j}$$

2.3.2 Information Filter

The information filter [6] (inverse covariance filter) is an algebraic reformulation of the Kalman filter. Here the estimated covariance and estimated state are substituted by the information matrix and the information vector respectively.

The covariance matrix 'P' represents uncertainty. The diagonal values of the matrix is directly proportional to the uncertainty. On the other hand, the diagonal values of the inverse of the matrix 'P' is inversely proportional to the uncertainty. This means when most valuable information is obtained, uncertainty decreases. That's the reason why inverse of covariance matrix is called as information. As the filter propagates this matrix and its respective vector it is called as an information filter.

These are represented as

$$Y_{k|k} = P^{-1}_{k|k}$$
$$y_{k|k} = P^{-1}_{k|k} x_{k|k-1}$$

Likewise, the predicted covariance and state have equivalent information forms, defined as:

$$Y_{k|k-1} = P^{-1}_{k|k-1}$$
$$y_{k|k-1} = P^{-1}_{k|k-1} x_{k|k-1}$$

These predicted information matrix and the information vectors are updated in the update step of this information filter.

For the update step, measurement matrix and measurement vector are required which are calculated as follows

$$I_k = H^T_K R^{-1}_k H_k$$
$$i_k = H^T_K R^{-1}_k z_k$$

For time invariant systems the transition matrix A, mapping matrix H, and measurement noise covariance matrices Q and R are constant matrices with respect to time.

Now, in the update step these predicted information matrix and the information vector are updated by adding the respective measurement matrix and the measurement vector.

$$Y_{k/k} = Y_{k/k-1} + \Sigma^N_{j=1} I_{K,j}$$
$$y_{k/k} = y_{k/k-1} + \Sigma^N_{j=1} I_{K,j}$$

The simple additive nature of the update stage makes the Information filter highly attractive for multi-sensor, decentralised and distributed estimation.

From the below equations we can calculate the error covariance matrix and the estimated state using the inverse operations.

$$P_{k|k} = Y^{-1}_{k|k}$$
$$x_{k|k} = Y^{-1}_{k|k} y_{k|k}$$

2.4 Cryptography

The field of cryptography[7] has vastly expanded, now it includes much more secret communication, such as, message authentication, protocols for exchanging secret keys, authentication protocols, electronic auctions, digital signatures and digital cash. In fact, modern cryptography is to be concerned with problems that may arise in any distributed computation that may come under internal or external attack.

2.4.1 Attack Scenarios

Few attack scenarios surface across, while the data is being communicated over secure/insecure channels by the server, For example:
Cipher-text only attack: This is the most generic and frequent attack, which refers to the scenario where the adversary just notices a cipher-text and attempts to determine the encrypted plain text.
Chosen plain text attack: The attacker has the possibility to acquire the encryption of any plain text of its choice. It then attempts to determine the plain text that was encrypted to return some different cipher text.
Chosen cryptic text attack: The attacker seeks to determine the plain text that was already encrypted to give a false cipher text, which prevents the attacker to obtain the decryption explicitly.

It is essential to be aware of the security notions that the scheme satisfies, before choosing a relevant encryption scheme, below are a few encryption schemes which could be employed in our distributed sensor network to preserve the privacy of the individual sensors:

- **IND CPA(Indistinguishability under Chosen Plain text Attack)**: Encryption scheme meeting IND-CPA allows the attacker to choose the plain text data to be encrypted, provided the attacker acquires no additional information about plain text data when the attacker learns its encryption.
- **AO(Aggregator Obliviousness)**: In AO the aggregator only learns the aggregate value from the encrypted values of n users at each time step.

Some schemes that we consider to employ and which satisfy the above security notions are: Paillier encryption scheme (IND-CPA), and Joye Libert private aggregation scheme (AO).

2.4.2 Paillier Encryption Scheme

Paillier encryption scheme is an asymmetric public key cryptography scheme, which uses two keys, generated using the key generation function by choosing two adequately large prime numbers(p,q). The product(N) of the prime numbers (p.q) is computed. A random number(g) is generated, where g = N+1. The bit length of p and q is equal. A public key (N,g), and a private key (p,q) w.r.t Public Key are obtained.

Encryption of plain text is done using a public key, and is computed as :

$$c = g^m r^n (mod N)^2$$

where :
r = random number and r^n = noise

We calculate the decrypted message(m) using the equation:

$$m = [\frac{L(c^{\lambda}(mod(N^2))}{L(g^{\lambda}(mod(N^2))}] \quad mod(N)$$

where λ (random number) and L(u) are:

$$\lambda = lcm(p-1, q-1)$$
$$L(u) = (u-1)/N$$

The encrypted cipher text is further decrypted using a private key, which ideally matches the public key used for encryption.

Homomorphic operation is performed on multiple plain texts, After decryption of the plain text.

$$D(\xi(m1)\xi(m2)(mod(N^2)) = m1 + m2(modN)$$

We use this above-mentioned paillier additive homomorphic operation to aggregate the encrypted information from the sensor nodes and then decrypt it to obtain the aggregated plain information, which is sent to an encrypted information filter in order to localize and estimate the object for the next time step.

2.4.3 Joye-Libert private aggregation scheme

Joye libert private aggregation scheme is built on paillier encryption scheme. But unlike paillier encryption, a honest user has to generate n private keys and distribute to the users to encrypt the plain text. This scheme meets the AO security notion which implies that the aggregator has no knowledge of the individual sensor data except the aggregated cipher text.

The key generation is similar to paillier, where two large primes are selected and hashing function. $H : Z \rightarrow Z^*_{n^2}$ represents the public key (N,H). Using this public key, private keys are computed and sent to n users to encrypt their respective information. These secret keys are then aggregated to a single final key to perform the aggregation of the cipher texts which are received from the individual n users and then decrypt it. Even though the aggregator has n individual texts, the attacker can not guess which user has sent a particular information. It is also difficult to revert the cipher text for the attacker because there are n private keys which are also aggregated to a single final key.

3 Implementation

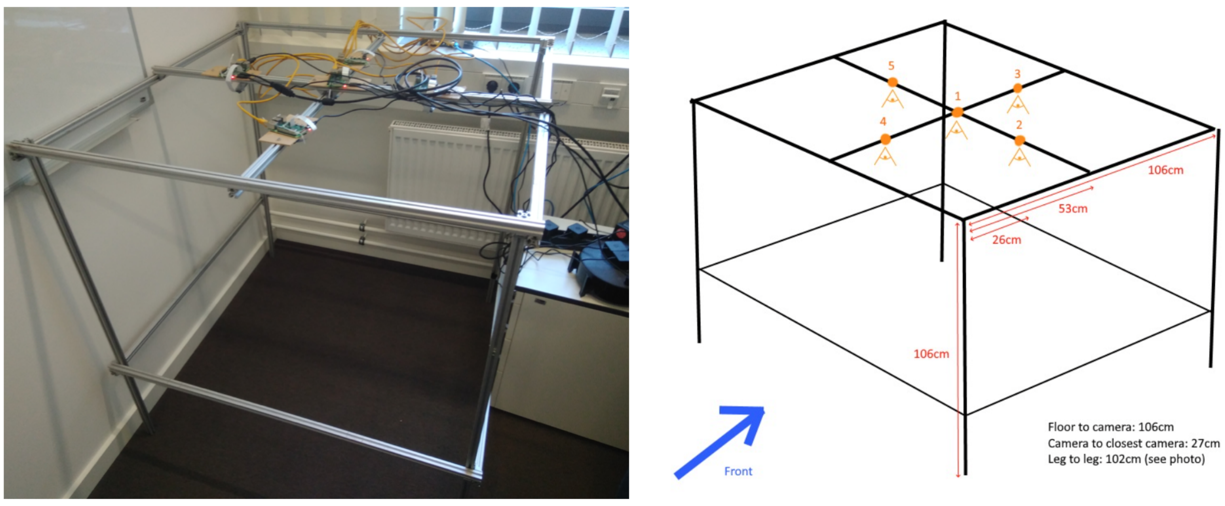
This section provides a detail description of how the project has been executed and also attempts to yield a feasible solution to our problem statements.

3.1 Hardware Setup

Hardware setup includes the physical and the functional components required to set up and conduct our experiment.

The list of components used are described below:

- Five Raspberry Pi-4 computers, 8GB Model-B with a 32GB class-10 Micro SD card, are rigidly fixed using screws at different positions on the metal frame.
- Raspberry Pi camera module Bus: CSI-2 Type with dimensions (23.86 x 25 x 9mm) are fixed to each individual Raspberry Pi.
- Ethernet Hub which connects and broadcasts signals to computers within a Local Area Network (LAN)
- Raspberry Pi imager installed in the hard disk of each Raspberry Pi, in order to create a Raspberry Pi OS.
- Red laser light(Object to be tracked).



(a) Frame **(b) Frame dimensions**

Figure 2: Frame architecture

The above figure 2 provides us with detail visual description of how the hardware and functional components are organised to complete our hardware setup, The Raspberry Pi modules are rigidly fixed at a distance of 106cm from the plane of subject (floor). The distance between each raspberry pi module is 26cm respectively. The leg to leg distance between the frame is 102cm. All Raspberry Pi-4 computers are connected to the same network (LAN).

3.2 Architectural Setup

The figure 3 describes our architecture setup, which includes information regarding roles assigned to each Raspberry Pi in our setup, the communication channel (client-server communication) and the python scripts[5]. These scripts manage individual sensors, local hub and the central node and their functionalities are explained in the following sub-sections 3.3, 3.4, 3.5 and 3.6.

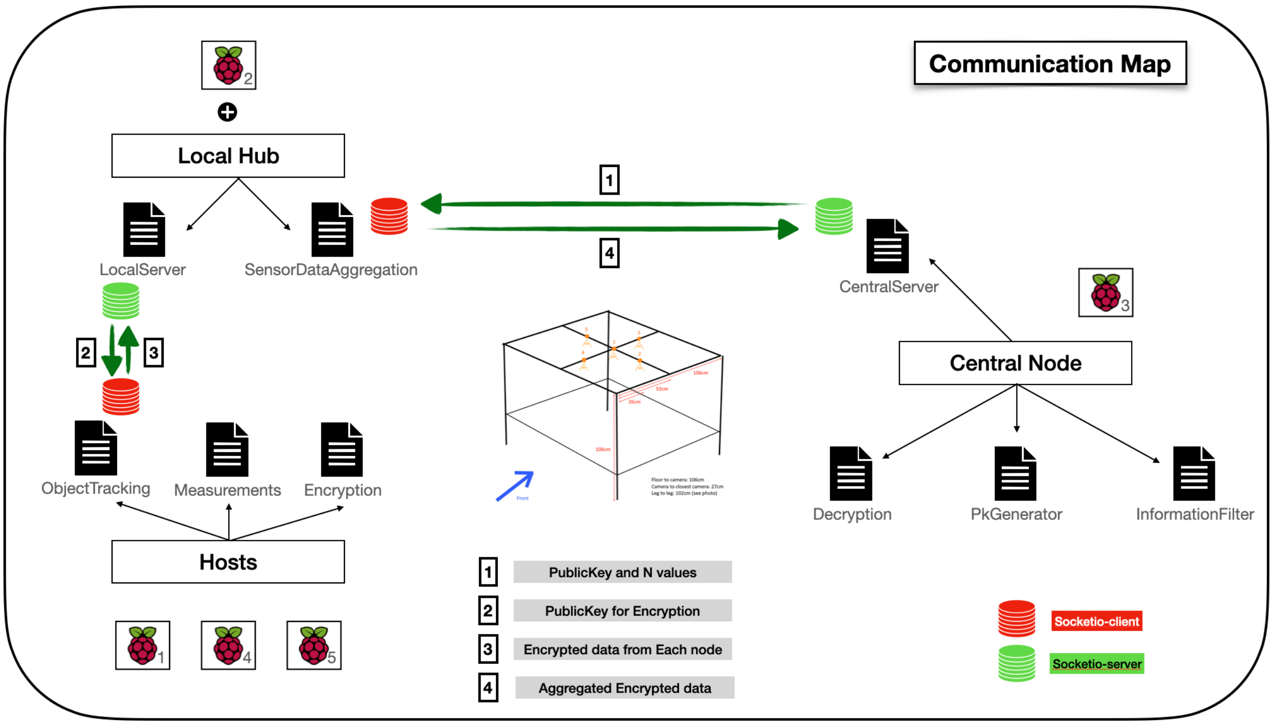


Figure 3: Communication layout of architectural setup

3.3 Object Tracking

The tracking algorithm was implemented in Python 3 using open source computer vision Library OpenCV. Considering the requirement of localisation of the objects for each sensor, we needed to grab the video frame by initialising an origin in the plane which is common to all the cameras facing towards it. Accordingly, We placed markers on the plane as reference to all the cameras, considering it to be the origin, we have initialised the the point by altering the pixels in x and y directions. We also considered perspective transformation as object and world coordinates are in 3D and the considered image coordinates are in 2D.

In order to detect the object we considered HSV colour space to perform red colour detection of the laser light in a dark background. We define lower mask and upper mask of the red colour in HSV space and any red pointer falling on the plane will be detected and a circle is drawn on the detected object for identification.

The next objective is to track the laser pointer on a 2D plane (for x, y coordinates). We know that the Raspberry pi camera module comes with a default configuration of 640 × 480p, we considered reference frame with specified dimension which allowed us to use a conversion factor with which we converted pixel to centimetres. This allowed us to use the coordinates generated while tracking the object to use as measurements later (Measurement vectors).

3.3.1 Measurement Vector

The sensors make the measurements 'z' which is a 2×1 vector consisting of x and y coordinates, these measurements are plugged in the equations mentioned in the information filter( ) and calculate the measurement vector of 4×1 dimension. We also calculate the measurement matrix using the mapping matrix 'H' of dimension 2×4 and the measurement noise covariance matrix 'R' of dimension 2×2. As these matrices 'H' and 'R' are public the calculated measurement matrix would be the same for all the sensor measurements

3.4 Networking

To establish communication between individual sensors and the central server we used TCP/IP protocol provided by python-socketio library in Python. Since the sensors are physically distributed in the same network and need to communicate to each other or a central node, we required distributed communication which we implemented by creating socketio client and server program, specifying their IP addresses.

We used this communication channel to initially transfer public key from the central node (consisting of decryption method) to local server which distributes the public key to each sensor used for encryption locally as an event. The encrypted data from each node is sent back to local server for aggregation. Once the aggregation is done at local server it sends the data to central server, to decrypt it and run the filter. As with any distributed environment, synchronisation is a critical aspect to be considered. We developed synchronous communication by sending acknowledgement to server once the received data is processed and requesting new set of data to be sent. We used synchronisation in two aspects:

- The local server waits until all the nodes are running parallel, in order to transmit the sensor readings every single time step synchronously.
- The central server after receiving the aggregated data, sends acknowledgement to local server to wait until processing is done, before sending next set of data.

The above followed steps also ensure the synchronisation at every single node.

3.5 Cryptography

In this project, the key motive is to maintain the privacy of the sensor data (measurements) from the untrusted parties. Hence, we applied the Paillier encryption scheme to preserve the privacy of the members. This scheme follows the following steps.

3.5.1 Key Generation

Both the operations ‘encryption’ and ‘aggregation’ require a public key which must be shared by the trusted third party, which decrypts the encrypted data. Hence, a public key and a private key for encryption and decryption must be generated before encrypting the information. Therefore, we use the ‘paillier key generation’ function from the ‘phe’ library to generate a public key and private key with sufficiently large primes ‘p’ and ‘q’ by setting n (length) to ‘128’ bits. The Raspberry Pi which acts as a ‘central node’ which generates the keys and distributes the public key to all the nodes.

3.5.2 Encryption and Aggregation

The measurement vectors are calculated and encrypted at the individual sensor using the function “Encryption”. Paillier encryption method from the paillier library. Once the measurement vectors are encrypted at each raspberry pi, the encrypted measurement vectors are sent to a local hub (Raspberry Pi). This local hub then aggregates each element of the respective sensor’s encrypted measurement vector homomorphically using paillier homomorphic operation, where the ciphertexts from the sensors are multiplied.

3.5.3 Decryption

The aggregated cipher text, received from local hub, is transferred to the central node for decrypting the aggregated cipher text with paillier decryption function ’decrypt’ using the private key which matches with the public key generated initially by the central node. As soon as the aggregated decrypted measurement information is obtained, this information is provided to the information filter.

3.6 Information Filter

To estimate the state of the laser pointer which follows a linear constant velocity model, we use an information filter which is run by the central node.

In this information filter[1], we have chosen state transition matrix ‘A’ to be of 4×4-dimension, initial error covariance matrix ‘P’ of dimension 4×4 and process noise covariance ‘Q’ to be of dimension 4×4. We initialize the state with [0,0,1,1]. The decrypted measurement vector is passed as an argument in the updation step of the information vector to give the new state estimate. We simply aggregate the measurement matrices in the updation step of the information matrix from which we can get the error covariance by applying inverse operations. The updated error covariance matrix 'P' and the estimated state '$x_{k/k}$' will be passed to the predict function for the next time step. These estimated states are appended to an array "updatedArray" to plot the graph.

Now that the state is estimated the central node sends the acknowledgement to the local hub to send the aggregated information of the next time step. After receiving the acknowledgment, the local hub now sends an acknowledgement to all the nodes to make the measurements, encrypt the information and send it to the local hub. In this way, synchronization is maintained.

A 'plot_graph' function is used in the information filter to plot the estimated states. We have used the "plt.savefig" command to save the plot in the mentioned path.

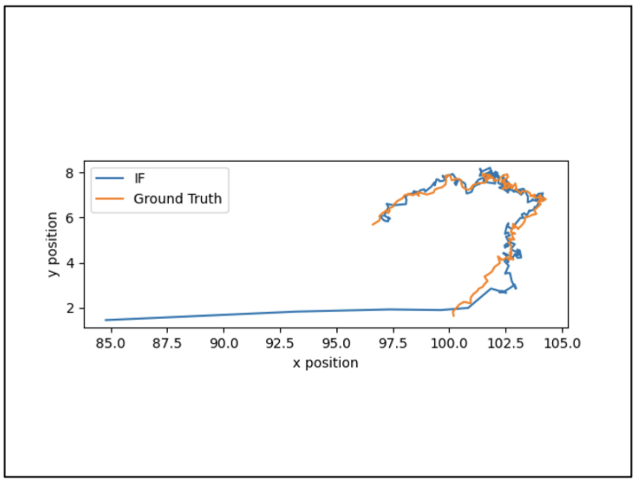
4 Evaluation and Results

As a part of evaluation, we simulated an encrypted information filter, by generating measurements from the process model and the measurement model. Because we have the ground truth in the simulation, it is easy to initially evaluate the proposed encrypted information filter. We assume that these measurements are made by the four sensors and these measurements are converted to measurement vectors. These measurement vectors are encrypted element wise and sent to a local hub which is another python instance assumed to be one of those sensors. These measurement vectors are aggregated using homomorphic addition operations using the public key given by the central node. The aggregated measurement vectors are then given to the information filter, where the measurement vector is updated in the filter update step and improves the estimates over time which can be seen in the error plot.

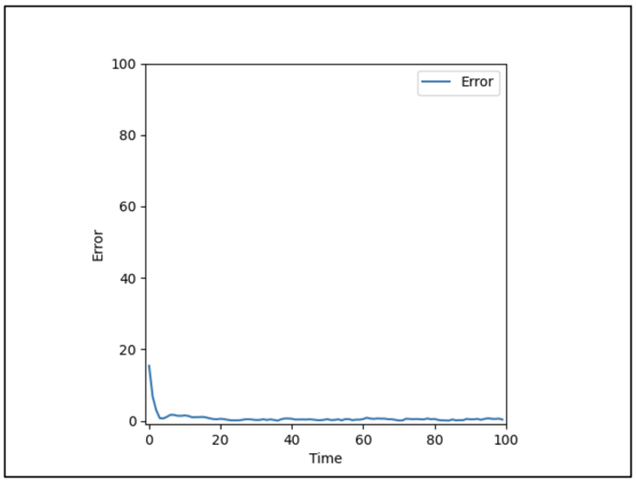
Two plots, one to show the error between the ground truth and the estimate 4 is plotted and the second which shows the estimates over time and ground truth 4. In addition to this, we tested the filter by giving very bad initial estimates and made it run. Initially the error was huge as the ground truth is far different from the initial estimate but with the time the error was minimal.

The results of our experimental setup, when we ran the filter on 4 nodes with random movement of laser pointer, is given in below plot 5

We also ran the setup on a fixed path trying to replicate linear constant velocity model following the line shown in the figure 6 produced the result plotted in the figure 6.



a. Estimated state (vs) Ground Truth



b. Error Plot

Figure 4: Simulation plot

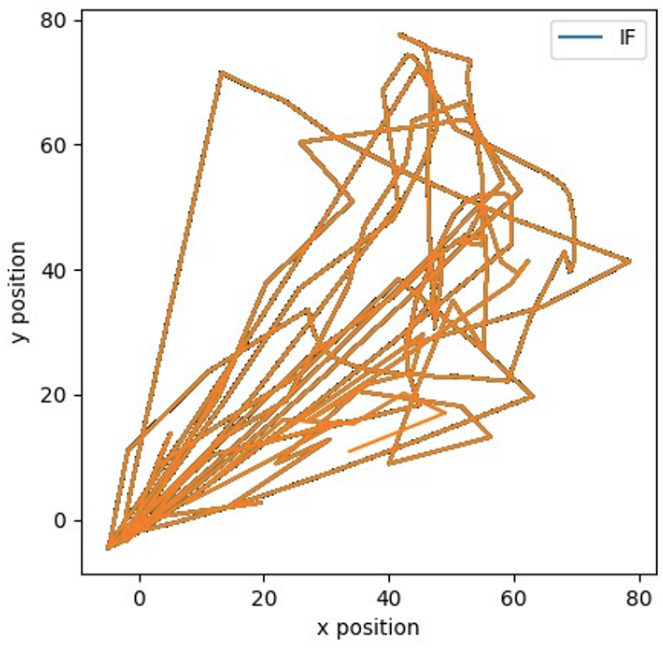


Figure 5: Plot of a randomly pointed laser on the plane

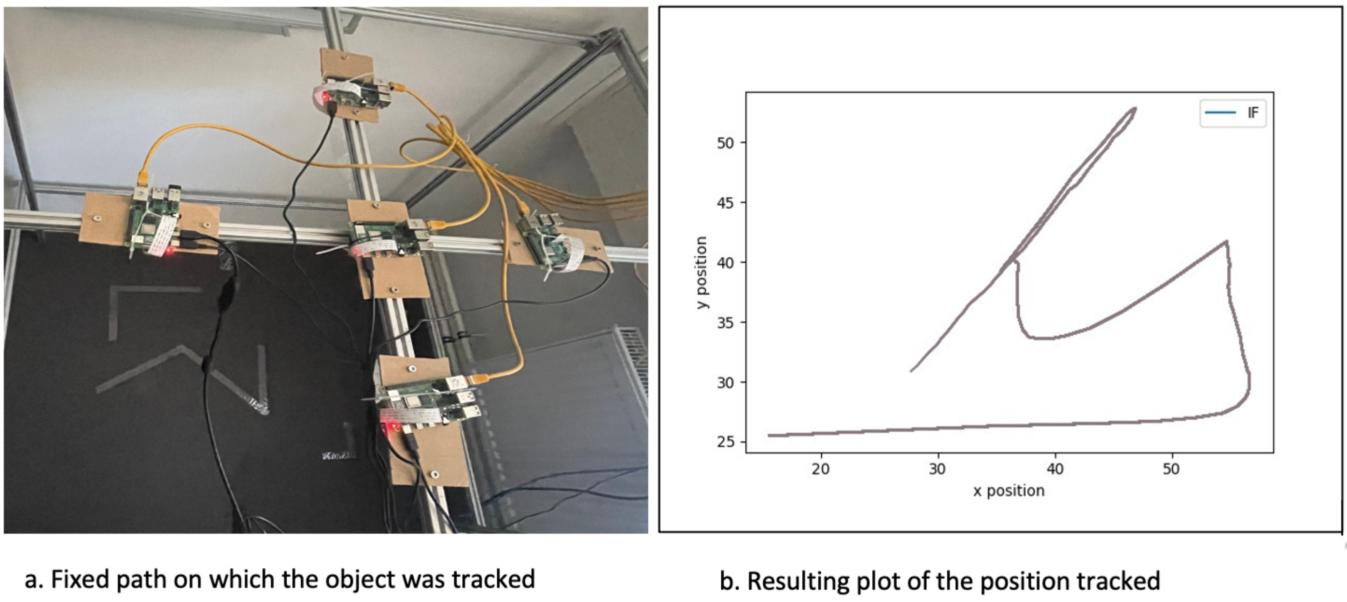


Figure 6: Plot for the fixed path

5 Conclusion

In this work, we provide a solution to preserve the privacy of the sensor data (measurements) and restrict the sensor data leakage by implementing a Paillier encryption scheme which can satisfy the security notion IND-CPA where if an attacker chooses a measurement vector of a particular node to be encrypted, then the attacker can only learn its ciphertext but cannot gain any additional information about the measurement vector as the length of the ciphertext is the same.

When each node encrypts the information and sends it to the local hub to aggregate the data, aggregation is performed after receiving the encrypted information from all the nodes. However, if there is an intervention of an unauthorized party which can have information about the individual sensor data before aggregation, privacy of the individuals is destroyed which concludes that the AO notion is not satisfied. Hence, it is important that a private aggregation scheme such as LCAO or Joye- Libert private aggregation scheme can be a good choice to strengthen the security which can be considered as a future work.

References

[1] Mikhail Aristov, Benjamin Noack, Uwe D Hanebeck, and Jörn Müller-Quade. Encrypted multisensor information filtering. In *2018 21st International Conference on Information Fusion (FUSION)*, pages 1631–1637. IEEE, 2018.

[2] Marko Ristic Benjamin Noack and Uwe D.Hanebeck. "privacy-preserving localization using private linear-combination aggregation".

[3] Kwassi H Degue and Jerome Le Ny. On differentially private kalman filtering. In *2017 IEEE Global Conference on Signal and Information Processing (GlobalSIP)*, pages 487–491. IEEE, 2017.

[4] A. Milanovic, S. Srbljic, and V. Sruk. Performance of udp and tcp communication on personal computers. In *2000 10th Mediterranean Electrotechnical Conference. Information Technology and Electrotechnology for the Mediterranean Countries. Proceedings. MeleCon 2000 (Cat. No.00CH37099)*, volume 1, pages 286–289 vol.1, 2000.

[5] Thalluri Nakka, Kanumuri. `https://code.ovgu.de/mushunur/pploc/-/tree/master/PPLOC_FinalCode`, 2021.

[6] Simon Parsons. Probabilistic robotics by sebastian thrun, wolfram burgard and dieter fox, mit press, 647pp., $55.00, isbn 0-262-20162-3. *Knowl. Eng. Rev.*, 21(3):287–289, September 2006.

[7] Marko Ristic, Benjamin Noack, and Uwe D. Hanebeck. Cryptographically privileged state estimation with gaussian keystreams. *IEEE Control Systems Letters*, 2021.

[8] Dan Simon. *Optimal state estimation: Kalman, H infinity, and nonlinear approaches*. John Wiley & Sons, 2006.