Modeling and Control for Cyber-Physical Systems

Project Activity I

1st Alessio Giorgetti
Politecnico di Torino
Matricola: 345637
S345637@studenti.polito.it

2nd Anna Roma
Politecnico di Torino
Matricola: 345819
anna.roma@studenti.polito.it

3rd Stefano Giulianelli
Politecnico di Torino
Matricola: 347911
stefano.giulianellli@studenti.polito.it

***Abstract*—This project focuses on secure state estimation and target localization/tracking in systems affected by sparse sensor attacks, situations where only a few sensors provide incorrect data due to intentional tampering. The main goal is to estimate the true state of the system and detect which sensors have been compromised. We implement and test several algorithms to prove that sparse optimization methods can successfully handle sensor attacks, offering accurate and robust performance even in challenging scenarios.**

I. Introduction

Cyber-Physical Systems (CPS) combine physical components (like sensors and actuators) with software and computation to monitor and control real-world environments. These systems are used in everyday technologies such as smart buildings, autonomous vehicles, and industrial automation. Because they rely heavily on sensor data, they are vulnerable to malicious attacks, especially when an attacker manipulates a few sensor readings to mislead the system.

In this project, we explore how to estimate the state of a system and track a moving target even when some sensors are giving faulty or manipulated data. We apply and compare different algorithms—such as ISTA, IJAM, SSO, D-SSO, and DISTA—that are designed to be robust against this kind of interference. These methods are based on sparse optimization, a mathematical approach that helps isolate and correct the effects of a small number of faulty measurements.

II. Task 1: Secure state estimation of a static CPS with sparse sensor attacks

Problem formulation

The first task proposed focuses on estimating the state of a CPS in the presence of sparse sensor attacks. The system is described by the following model:

$$y = C\tilde{x} + \tilde{a} + \eta \tag{1}$$

where:

- $\tilde{x} \in \mathbb{R}^n$: unknown state vector,
- $\tilde{a} \in \mathbb{R}^q$: unknown sparse attack vector,
- $\eta \in \mathbb{R}^q$: measurement noise.

The aim is to estimate both the state of the system and the support of the attack vector, while tuning the algorithmic parameters to improve the performance. In other words, our goal is to recover $\tilde{x}$ and identify the nonzero entries of $\tilde{a}$, corresponding to the sensors under attack.

A key tool in both ISTA and IJAM is the **soft-thresholding operator**, defined componentwise for $z \in \mathbb{R}$ and threshold $\lambda > 0$ as:

$$S_\lambda(z) = \text{sign}(z) \cdot \max(|z| - \lambda, 0). \tag{2}$$

This operator promotes sparsity in the solution by forcing small values to zero and reducing higher values of the quantity λ.

In this task, we implement and compare two algorithms: **ISTA** and **IJAM**:

- **ISTA** performs a gradient descent step on x (since the loss function F is differentiable) and applies a soft-thresholding (proximal mapping) on a to promote sparsity.
- **IJAM** alternates between solving for x via pseudo-inversion (fixing a) and applying soft-thresholding on a (fixing x).

The key difference lies in how x is updated: ISTA uses gradient descent, while IJAM uses least-squares updates. ISTA is generally easier to tune and more flexible, while IJAM can converge faster in certain settings.

Experimental results

The experiment was set up using $n = 15$ states, $q = 30$ sensors, and $h = 2$ active sensor attacks. We set the regularization parameter $\lambda = 0.1$, and used appropriate step sizes ν for each algorithm: $\nu = 0.7$ for IJAM and $\nu = \frac{0.99}{\|G\|_2^2}$ for ISTA, with $G = [C\ I]$. The stopping condition was based on a threshold $\delta = 10^{-10}$. Each configuration was tested over 20 runs, both algorithms were stopped when the differences between the last two iterations were smaller than a fixed small δ and the final results were averaged.

We started with the alternating minimization of the IJAM algorithm, at every iteration we fix a and minimize the P-Lasso of the problem with respect to x by pseudo-inversion. Then we fix x and minimize with respect to a by using the proximal mapping. The Table [I] shows, from the State Estimation Error derivation, that both algorithms are relatively accurate in estimating the true state of the system, correctly finding the true value of the support of the attack vector, as we can see in the image [5] below.

Error	IJAM	ISTA
State estimation error	0.005393 $\pm$ 0.001834	0.005546 $\pm$ 0.001865
Support attack error	0.10 $\pm$ 0.0.31	0.15 $\pm$ 0.49

TABLE I: Performance comparison of IJAM and ISTA

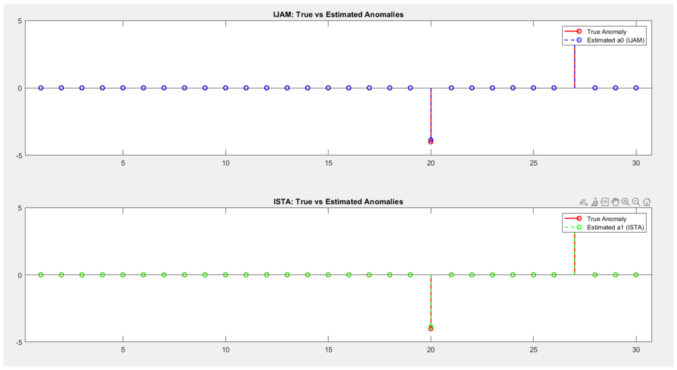

Fig. 1: Found support attack vector

The graph [2] shows the convergence rate of ISTA (blue line) compared with IJAM (red line). In the initial iterations, the ISTA algorithm exhibits cleaner and more stable behavior compared to IJAM. However, in the long term, IJAM converges significantly faster to a value that closely approximates the true state of the system.

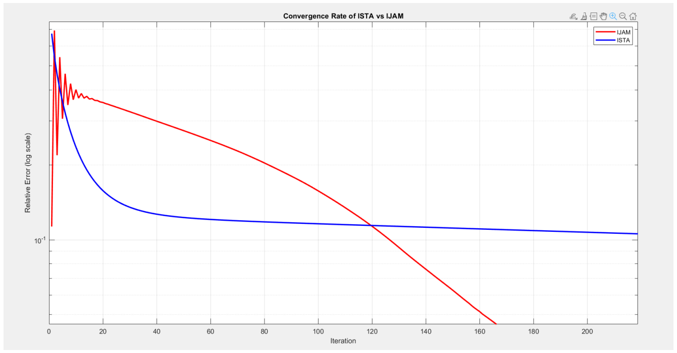

Fig. 2: Convergence rate

To better interpret the performance of the algorithms, we now analyze how the choice of hyperparameters λ and ν influences the results.

Error	IJAM	ISTA
State estimation error	0.062078	0.062201
Support attack error	0.05 $\pm$ 0.22	0.05 $\pm$ 0.22

TABLE II: Performance comparison of IJAM and ISTA

We first modify the regularization parameter λ, testing two more values: $0.05$and 1. As shown in Tables [III], [I], and [II], $\lambda = 0.1$ provides the best trade-off between state estimation accuracy and correct detection of attacked sensors.

Error	IJAM	ISTA
State estimation error	0.003453	0.003677
Support attack error	0.20 $\pm$ 0.41	0.40 $\pm$ 0.68

TABLE III: Performance comparison of IJAM and ISTA

Increasing λ to 1 improves sparsity and reduces false positives, but worsens state estimation accuracy. In contrast, decreasing it to 0.05 improves accuracy, but increases sensitivity to noise and false detections. In summary, higher λ values lead to faster but less precise convergence, while lower values offer better accuracy at the cost of stability and robustness.

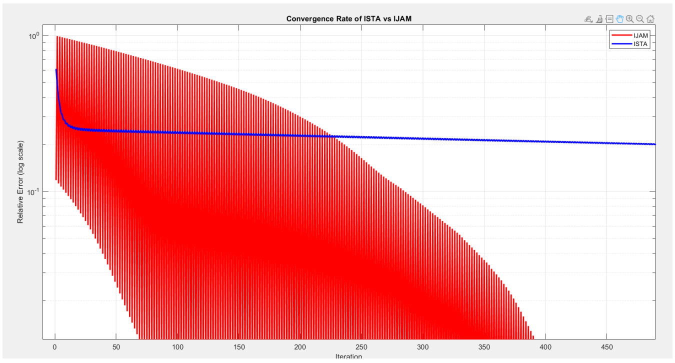

Fig. 3: Convergence rate with ν increased

We then evaluate the effect of changing the step size ν: by increasing ν to 1 for IJAM and to $2/\|G\|_2^2$ for ISTA, Figure [3] shows faster convergence. However, this increased aggressiveness in optimization reduces the precision of the final estimates. On the other hand, reducing ν to 0.3 for IJAM and $0.5/\|G\|_2^2$ for ISTA improves estimation accuracy but slows down the convergence significantly, as illustrated in Figure [4]. In this case, both algorithms are now more accurate, but this comes at the cost of a significantly slower convergence speed compared to before.

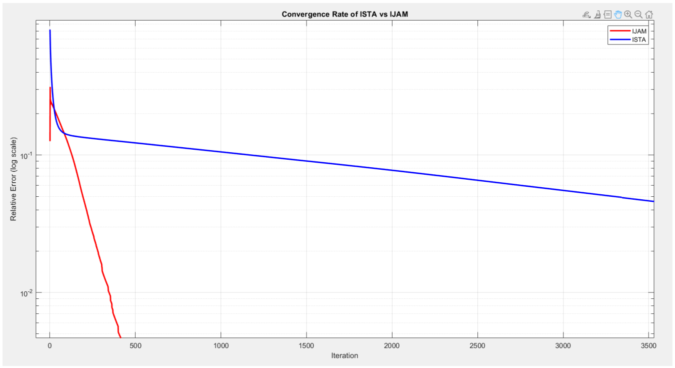

Fig. 4: Convergence rate with ν decreased

Finally, we stress-test the system by increasing the number of attacked sensors from $h = 2$ to $h = 5$.

As shown in Table IV, increasing the number of attacked sensors significantly degrades performance. The state esti-

Error	IJAM	ISTA
State estimation error	0.051786	0.075934
Support attack error	3.60 ± 4.26	6.75 ± 4.93

TABLE IV: Performance comparison of IJAM and ISTA

mation and the support attack errors become particularly significant due to the increased difficulty in isolating multiple corrupted sensors. The higher complexity makes accurate state estimation more difficult, with a notable rise in false positives in attack detection. This reduced resilience is clearly illustrated in Figure [5], where incorrect detections and false positives become more frequent. It is clear that with more sensors under attack, the complexity of the problem increases, leading to less accurate estimations. The attack support estimation is the most affected, with false positives and incorrect estimations by the algorithm. This is evident in the following plot, where we can clearly observe the impact:

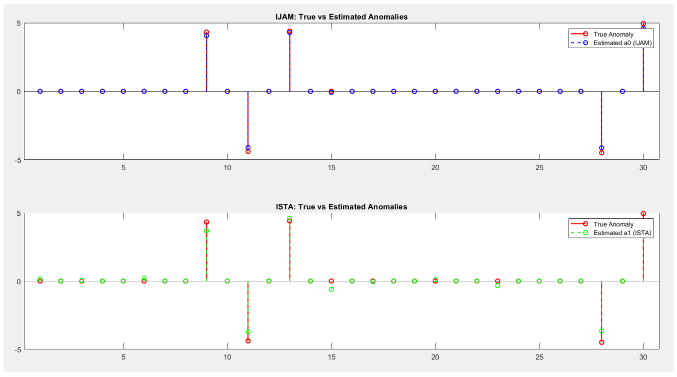

Fig. 5: Found support attack vector

Over the 20 averaged trials, both IJAM and ISTA demonstrated high accuracy, with ISTA being more stable initially and IJAM converging faster over time. Varying λ and ν revealed a trade-off between convergence speed and estimation precision: larger values favor speed and sparsity, while smaller ones improve accuracy but increase sensitivity to noise and computation time. Increasing the number of attacked sensors reduced estimation reliability and led to more false positives, highlighting the system's sensitivity to sparsity violations.

III. Task 2: Target localization under sparse sensor attacks

Problem formulation

The localization of a target in an indoor environment can be framed into three main tasks: **detection** refers to determining whether the target is present in the monitored area; **localization** involves estimating the static position of the target; **tracking** concerns the dynamic estimation of the target's trajectory over time. In this project, we focus on the **localization** task using a *fingerprinting-based* approach with RSS (Received Signal Strength) measurements.

The *fingerprinting method* consists of two phases: an offline training phase and an online runtime phase. In the training phase, the environment is divided into n discrete cells and q sensors are deployed (see Figure [6]). The RSS values are collected in a dictionary matrix $D \in \mathbb{R}^{q\times n}$ when the target is located in each cell, such that each column D_j corresponds to the RSS pattern observed, when the target is located in cell j.

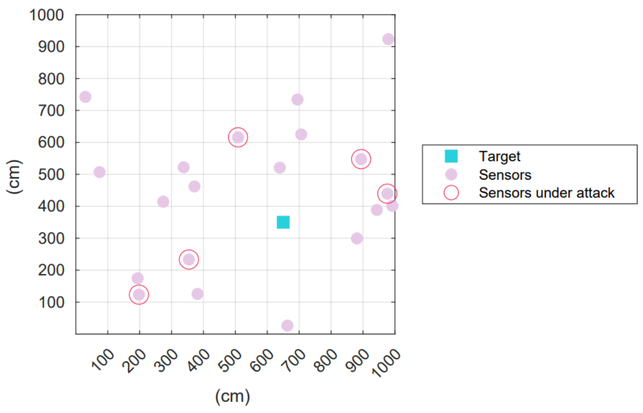

Fig. 6: Fingerprinting-based localization problem

In the online phase, the sensors collect a measurement vector $y \in \mathbb{R}^q$, and our goal is to estimate the current position of the target by comparing y with the stored fingerprints in D. In adversarial scenarios, some sensors may be compromised, reporting corrupted values. The measurement model is defined by the equation:

$$y = Dx + a, \tag{3}$$

where $x \in \mathbb{R}^n$ is the sparse vector indicating the target's position and $a \in \mathbb{R}^q$ is the sparse attack vector. The problem is now a weighted Lasso:

$$\min_{x\in\mathbb{R}^n,\, a\in\mathbb{R}^q} \left\| G\begin{pmatrix} x \\ a \end{pmatrix} - y \right\|_2^2 + \lambda_1\|x\|_1 + \lambda_2\|a\|_1, \tag{4}$$

where G is the design matrix with zero-mean and unit-variance columns and λ_1, λ_2 are regularization parameters promoting sparsity in both the position and the attack vectors.

We solve this optimization problem using ISTA algorithm defined in Fig [4], because this method allows us to robustly estimate both the location of the target and the set of attacked sensors, even in the presence of adversarial tampering.

Algorithm 1 ISTA Algorithm

```
Require: $x^{(0)} = 0 \in \mathbb{R}^n$, $a^{(0)} = 0 \in \mathbb{R}^q$
1: for $k = 0, 1, 2, \ldots, T_{\max}$ do
2:     $z^{(k)} = \begin{bmatrix} x^{(k)} \\ a^{(k)} \end{bmatrix} - \nu G^\top \left(G \begin{bmatrix} x^{(k)} \\ a^{(k)} \end{bmatrix} - y\right)$
3:     $x^{(k+1)} = \mathcal{S}_{\nu\lambda}\left(z^{(k)}_{1:n}\right)$
4:     $a^{(k+1)} = \mathcal{S}_{\nu\lambda}\left(z^{(k)}_{n+1:n+q}\right)$
5: end for
```

Experimental results

The algorithm generates 2 sparse matrices that describe the evolution of the vectors at each iteration: $x \in \mathbb{R}^{nxT}$ is a sparse vector indicating the target's position ($\tilde{x}_{i,j} \neq 0 \iff target\ is\ in\ cell\ i\ at\ iteration\ j$), and $a \in \mathbb{R}^{qxT}$ is a sparse attack vector affecting ($\tilde{a}_{i,j} \neq 0 \iff sensor\ number\ i\ was\ attacked\ at\ iteration\ j$. Our goal is to calculate the final estimation of the attacks received and the final target's position, which are the last columns of the two matrices.

The ISTA algorithm was tested on a scenario with one true target and results are shown in first raw of Table [V].

λ	ν	supp($\tilde{x}$)	supp($\tilde{a}$)
10	0.0014	[10 37 59]'	[1 10 14 16 17]'
90	0.0014	[10 37]'	0×1 empty double column vector
75	0.0014	[10 37 59]'	[16 17]'
10	0.0028	[10 37 59]'	[1 10 14 16 17]'
10	2.8388e-04	-	-
10	0.0142	-	-

TABLE V: Tuning ISTA parameters for localization

The estimated support of the signal vector $\tilde{x}$ included three active positions, indicating that the algorithm overestimated the number of targets. This suggests that, under the influence of sparse sensor attacks, the method struggles to correctly isolate the true target and instead introduces false positives. To address this, we experimented with tuning of the regularization parameter λ (the step size) and ν. Increasing the regularization parameter λ significantly to 90 reduces the number of false positives in the support of $\tilde{x}$, but the algorithm completely fails to estimate the attack vector $\tilde{a}$. With an intermediate value of $\lambda = 75$, a smaller attack vector is recovered (though not all attacks are identified), and the support of $\tilde{x}$ still includes false positives. On the other hand, tuning only the step size ν leads to unstable results: for very small values, the algorithm fails to produce any meaningful estimate, while for larger values, convergence is still not achieved and the quality of the solution remains poor.

In conclusion, the ISTA algorithm struggles to accurately recover the true target support in presence of sparse sensor attacks. Even after tuning the parameters, the algorithm fails to identify the attack vector and state vector support. These results suggest that ISTA, while simple and computationally efficient, is not sufficiently robust for this type of adversarial localization problem.

IV. Task 3: Secure state estimation of a dynamic CPS with sparse sensor attacks

Problem Formulation

For the dynamical system, we consider the following model, with $q = 30\ sensors$, $n = 30\ state\ dimension$ and $h = 3\ sensors\ under\ attack$:

$$\begin{cases} x(k+1) = Ax(k) \\ y(k) = Cx(k) + a(k), \end{cases} \tag{5}$$

where $x(k) \in \mathbb{R}^n$ is the system state, $y(k) \in \mathbb{R}^q$ is the measurement vector, $A \in \mathbb{R}^{n\times n}$ and $C \in \mathbb{R}^{q\times n}$ are known system matrices, and $a(k) \in \mathbb{R}^q$ is a sparse attack vector. The number of attacked sensors is assumed to be much smaller than the total number of sensors, i.e., $h \ll q$. Applying a deadbeat approach and collecting T successive measurements,we build an augmented system with the assumption that the attack is constant over time, i.e., $a(k) = a$, the attack support remains fixed during a reasonable time window.

This results in a system of qT equations with $n + q$ unknowns. We can write:

$$\begin{pmatrix} y(0) \\ y(1) \\ \vdots \\ y(T-1) \end{pmatrix} = \mathbb{O}_T x(0) + \begin{pmatrix} a \\ \vdots \\ a \end{pmatrix} = \underbrace{\begin{pmatrix} C & I \\ CA & I \\ \vdots & \vdots \\ CA^{T-1} & I \end{pmatrix}}_{\mathbb{O}'_T} \begin{pmatrix} x(0) \\ a \end{pmatrix} \tag{6}$$

We first ask if the classic Luenberger observer could be a good strategy to compute the problem with the augmented system. In order to do that, we apply Theorem [IV] and calculate the eigenvalues of matrix A (Tab. [VI]).

1.0000 + 0.0000i
0.1936 + 0.7374i
0.1936 - 0.7374i
0.4672 + 0.4928i
0.4672 - 0.4928i
-0.8406 + 0.0000i
-0.4721 + 0.4433i
-0.4721 - 0.4433i
0.3461 + 0.1376i
0.3461 - 0.1376i
0.0044 + 0.3822i
0.0044 - 0.3822i
-0.4680 + 0.0000i
-0.1674 + 0.0888i
-0.1674 - 0.0888i

TABLE VI: Eigenvalues of matrix A

Theorem I: Observability under Constant Attacks
Let the pair (A, C) be observable in the absence of attacks. If the attack vector $a(k) = a$ is constant over time and the matrix A has one eigenvalue equal to 1, then the augmented system is **not** observable.

Since matrix A has one eigenvalues, the first one, equal to 1, we cannot apply the Theorem and we must conclude that the augmented system is not observable. More in general, cyber-physical systems (CPS) under sparse attacks are not observable, even when the attack is constant and its support is known. Therefore, traditional observers such as the Luenberger observer cannot be used. To handle sparsity and solve our problem, we consider two observer structures:

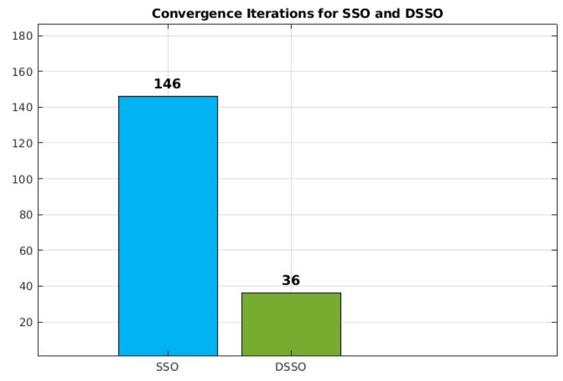

Fig. 7: Convergence Iterations SSO vs DSSO

- **SSO (Sparse Soft Observer)** [Algorithm 2]: estimates $x(k)$ using a Luenberger observer under the assumption that (A, C) is observable and estimates $a(k)$ using a soft-thresholding operator based on sparsity.
- **DSSO (Deadbeat Sparse Soft Observer)** [Algorithm 3]: similar to **SSO**, but with a more aggressive gain L, because it uses a deadbeat observer where L is designed such that $\text{eig}(A - LC) = 0$. This is achieved by computing $L = A * C^+ = A * (C^T C)^{-1} C^T$, where C^+ is the pseudoinverse of C, such that $A - LC = 0$.

Algorithm 2 SSO Algorithm

```
Require: $x^{(0)} = 0 \in \mathbb{R}^n$, $a^{(0)} = 0 \in \mathbb{R}^q$
1: for $k = 0, 1, 2, \dots, T_{\text{max}}$ do
2:     $x^{(k+1)} = Ax^{(k)} + L(y^{(k)} - Cx^{(k)} - a^{(k)})$
3:     $a^{(k+1)} = \mathcal{S}_\lambda(y^{(k)} - Cx^{(k+1)})$
4: end for
```

Algorithm 3 DSSO Algorithm

```
Require: $x^{(0)} = 0 \in \mathbb{R}^n$, $a^{(0)} = 0 \in \mathbb{R}^q$
1: Design $L$ such that $A - LC = 0$ ($L = A * C^+$)
2: for $k = 0, 1, 2, \dots, T_{\text{max}}$ do
3:     $x^{(k+1)} = Ax^{(k)} + L(y^{(k)} - Cx^{(k)} - a^{(k)})$
4:     $a^{(k+1)} = \mathcal{S}_\lambda(y^{(k)} - Cx^{(k+1)})$
5: end for
```

Experimental results

As a first step, due to the restriction imposed by Theorem [IV], we investigated whether SSO and DSSO would converge and at which iteration each algorithm would stop. In Figure [7], it is clear that both algorithms reach convergence, but SSO (blue line) is significantly slower (146 iterations) compared to DSSO (green line), which converges in only 36 iterations. This observation motivates further analysis to determine which algorithm performs better overall.

The plots in Figure [8] show the estimation errors for both the state and the attack vectors. For both estimations, SSO appears to be more accurate in the early iterations. However, as time progresses, DSSO surpasses SSO in performance, converging faster and yielding a lower estimation error. This behavior suggests that the choice between SSO and DSSO depends on the application requirements. If accurate estimation is needed within very few iterations (e.g., less than 30), SSO may be preferable. However, for long-term accuracy and faster convergence, DSSO is clearly the better choice—providing better estimates for both state and attack vectors already after 40 iterations.

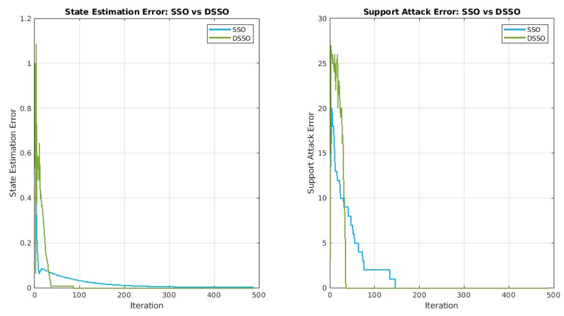

Fig. 8: State and Attack estimation errors SSO vs DSSO

V. Task 4: Target tracking under sparse sensor attacks

Problem formulation

Now, we address the problem of tracking a moving target in an indoor environment. Unlike Task 3, which focused on static localization, we now aim to estimate the real-time trajectory of the target, represented by the state vector $x(k)$. The problem also takes care of sparse adversarial attacks that affect a subset of sensors, introducing corrupted measurements in the received signal $y(k)$. The system is now modeled as:

$$\begin{cases} x(k+1) = Ax(k) \\ y(k) = Dx(k) + \tilde{a} + \eta \end{cases} \tag{7}$$

where:

- $x(k) \in \mathbb{R}^n$ is the state vector representing the target position at time k,
- $A \in \mathbb{R}^{n \times n}$ is the known state transition matrix,
- $y(k) \in \mathbb{R}^q$ is the vector of sensor measurements,
- $D \in \mathbb{R}^{q \times n}$ is the known measurement matrix,
- $\tilde{a} \in \mathbb{R}^q$ is the known sparse attack vector (assumed constant),
- η is the measurement noise.

Our goal is to estimate the true state trajectory $x(k)$ over time and identify the attacked sensors by recovering the support of $\tilde{a}$. Before that, it is necessary to analyze whether the system is observable, considering the pair (A, D) in the absence of attacks. Following the same calculation of the system 6 in Task 3, the observability matrix $\mathcal{O}$ is defined as:

$$\mathcal{O} = \begin{bmatrix} D \\ DA \\ DA^2 \\ \vdots \\ DA^{n-1} \end{bmatrix} \quad (8)$$

Following the **Kalman Method**, a system is **observable** $\iff \mathcal{O}$ is **full rank**. In our case, if $\text{rank}(\mathcal{O}) = n$ (dimension of the system), then the system in analysis is observable. This means that the initial state $x(0)$ can be uniquely reconstructed from the output sequence $\{y(k)\}_{k=0}^{T-1}$ (assuming $T \geq n$).

When considering constant attacks, the system can be rewritten in augmented form:

$$z(k) = \begin{bmatrix} x(k) \\ a \end{bmatrix}.$$

Then the dynamics become:

$$\begin{cases} z(k+1) = \begin{bmatrix} A & 0 \\ 0 & I \end{bmatrix} z(k) \\ y(k) = \begin{bmatrix} D & I \end{bmatrix} z(k). \end{cases} \quad (9)$$

Now, to see if the augmented system is still observable, it is again important to define a new observation matrix:

$$\mathcal{O} = \begin{bmatrix} D & I \\ DA & I \\ DA^2 & I \\ \vdots & \\ DA^{n-1} & I \end{bmatrix} \in \mathbb{R}^{qn \times (n+q)} \quad (10)$$

We notice that the columns associated with the attack vector $\tilde{a}$ are repeated between all rows, since the attack is constant. This leads to linear dependence among those columns (so clumns are not linear indipendent), which implies that the observability matrix cannot be not full rank. As a result, by applying Theorem [IV], the augmented system defined by Eq. [9] is **not** observable and this means that there are ∞ solutions to the problem. Therefore, a standard Luenberger observer cannot be applied in this setting. To handle sparse attacks and ensure robust tracking, we implement the **SSO** with a modification (w.r.t. Task 3, Alg. [4]): since the true state $x(k)$ is sparse (only one cell is occupied at a time and the number of cells is much larger than the number of targets), we apply soft-thresholding also to the state update. The matrix $G = [D\ I]$ is normalized to have zero mean and unit variance columns, which ensures that the regularization affects all components equally.

Experimental results

Figure [9] illustrates the estimation errors of both the state and attack vectors using the SSO algorithm. Regarding the attack vector, the observer converges rapidly, with the estimation error decreasing to zero within a few iterations. In contrast, the state estimation error converges to the value of 1 (nonzero) and has a consistent number of high impulse oscillations towards the end of the simulation. This behavior is similar to what was observed in Task 3, where the state estimation error was also nonzero.

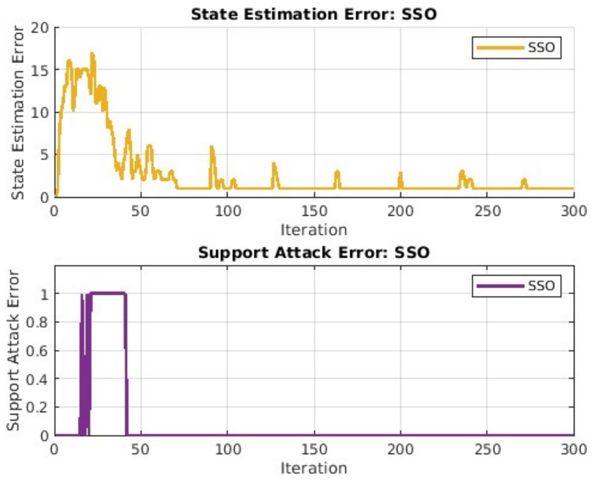

Fig. 9: Estimation errors while tracking using SSO

For this reason, we investigate whether applying **DSSO** in this context leads to improved performance. The DSSO algorithm, described in Alg. [5], exploits the sparsity and constancy of the attack vector while forcing a deadbeat dynamic through the choice of $L = AD^+$. Despite the lack of formal observability, we expect it to provide accurate estimates for both the state and the attack vectors in very few iterations.

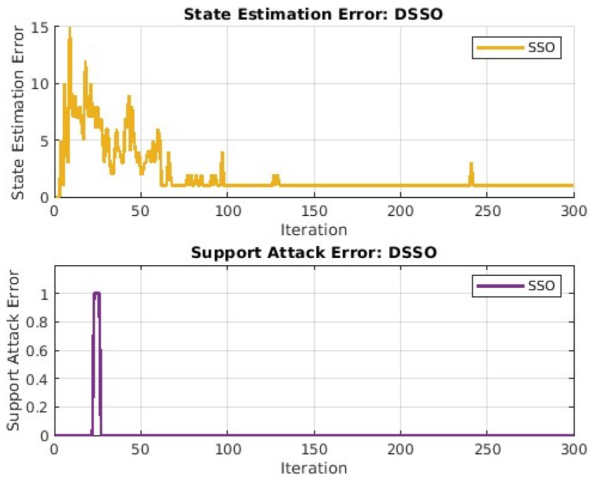

Fig. 10: Estimation errors while tracking using DSSO

Figure [10] shows the estimation errors for DSSO, with a tuning parameter $\nu = 0.02$. The results indicate that both errors decrease smoothly. In particular, the state estimation error still converges to 1, but without oscillations, unlike SSO, and the attack estimation also reaches convergence to zero.

To conclude, Figure [11] shows the comparison of the

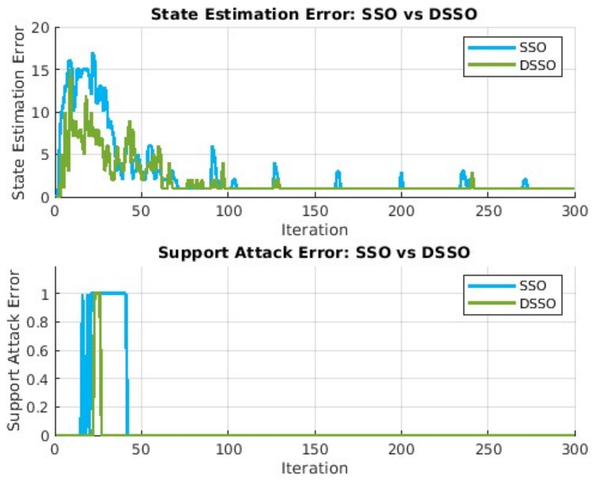

Fig. 11: Estimation errors while tracking using SSO vs DSSO

two methods, which highlights the important differences in performance.

The graph clearly illustrates that for large iterations, DSSO guarantees smoothness for the state estimation error and reaches convergence much faster for the attack estimation error, an important advantage over SSO. Therefore, we can conclude that DSSO is more reliable and efficient for Cyber Physical System than SSO.

VI. Task 5: Distributed target localization under sparse sensor attacks

Problem Formulation

We now return to **Task 2**, where we tried to solve the problem of localizing a target by estimating its position under sparse sensor attacks. In this case, we perform the same problem but on a distributed frameworks environment.

We consider a set of interconnected devices, referred to as **agents**, each collecting measurements through their own sensors. There are three different modes for data processing:

- **Centralized Processing:** All data is sent to a central Fusion Center, which performs all the processing.
- **Fully Distributed (In-Network) Processing:** Agents are interconnected and share information among themselves to compute results collaboratively, without a central Fusion Center.
- **In-Network with Fusion Center:** A hybrid approach in which agents exchange information and a Fusion Center computes the final estimate.

The system is modeled as a **multi-agent system**, which can be represented as a Linear Time Invariant (LTI) system of the form:

$$x(k+1) = Qx(k) \ ,$$

such that, each Agent i can be represented as:

$$x^{(i)}(k+1) = \sum_{i=1}^{q} Q_{i,j}x^{(j)}(k) \ = \ x^{(i)}(k)+\sum_{j\neq i} Q_{i,j}[x^{(j)}(k)-x^{(i)}(k)] \ , \tag{11}$$

were, Q is a **stochastic matrix** in which each raw row (that represents an Agent) sums up to 1 and our goal is to reach agreement acros all Agents. This formulation corresponds to a **Consensus Problem**: consensus is achieved when all rows of $Q^{(k)}$ (Q at iteration k) are equal, meaning all agents agree on a common value.

Sufficient condition to solve the Consensus Problem is to find the Convergence Rate by the essential spectral radius of Q, specifically the second-largest eigenvalue λ_2, such that $1 = |\lambda_1| > |\lambda_2| \geq \cdots \geq |\lambda_q| \ \lambda_1$. A higher λ_2 implies faster convergence.

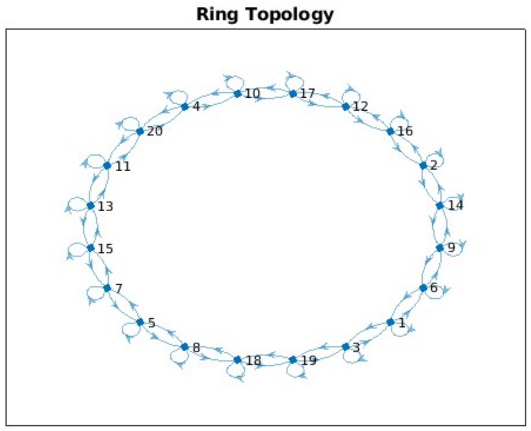

Fig. 12: Ring Topology

In our calculations we assume that the central node $z^{(0)}$ performs the mean of all the other nodes' states. To visualize the interconnection between the nodes in our problem, we consider a ring topology (Figure [12]) described by the matrix Q_{ring}, with eigenvalues vector of the form:

$$\begin{bmatrix}1.0000, \ 0.9674, \ 0.9674, \ 0.8727, \dots \ 0.0585\end{bmatrix} \ .$$

since the largest eigenvalue is $\lambda_1 = 1$ and second largest eigenvalue is $|\lambda_2| = 0.9674 < 1$, we expect a fast convergence.

Experimental results

To solve the consensus problem, we use **DISTA** (Distributed Iterative Shrinkage-Thresholding Algorithm), an algorithm that applies the soft-thresholding operator (defined in Equation [2]) in a distributed environment.

Algorithm 4 ISTA Algorithm

```
Require: $x^{(i)}(0) = 0 \in \mathbb{R}^n$
1: for $k = 0, 1, 2, \dots, T_{\max}$ do
2:     $x^{(i)}(k+1) = \mathcal{S}_{\lambda\tau}\left(\sum_{j=1}^{q} Q_{i,j}\, x^{(j)}(k) + \tau A_i^\top (y_i - A_i x^{(i)}(k))\right)$
3: end for
```

We also adopt two different topologies (ring topology in Figure [12] and star topology in Figure [13])and then compare the results, to better understand the dynamics of the algorithm.

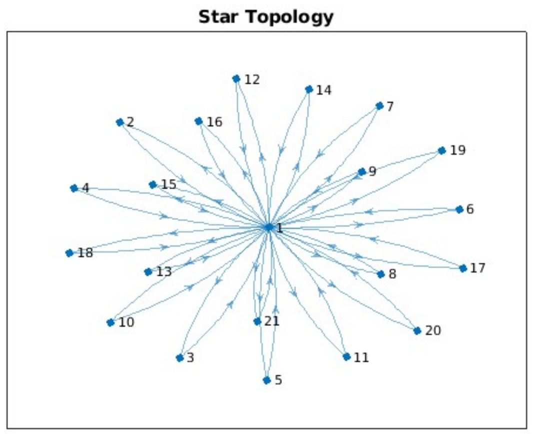

Fig. 13: Star Topology

From the problem formulation, we know that consensus is achieved only when the algorithm converges. Therefore, to verify convergence, we generate a bar plot that shows the number of iterations required to reach convergence.

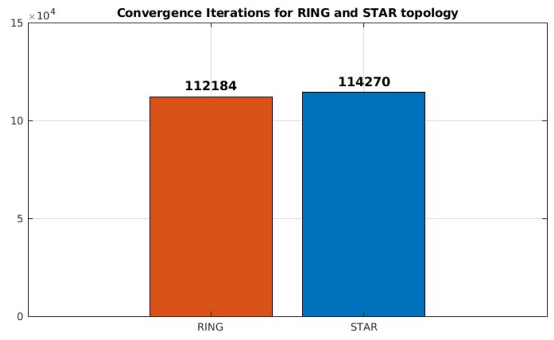

Fig. 14: Star VS Ring Topology convergence iterations

From the Figure [14], we observe that the ring topology (red bar) requires fewer iterations to reach consensus compared to the ring topology (blue bar). However, both algorithms at convergence reach the same support state vector and support attack vector:

Topology	supp($\tilde{x}$)	supp($\tilde{a}$)
Ring	37	[1 10 14 16 17]'
Star	37	[1 10 14 16 17]'

TABLE VII: Algorithm results

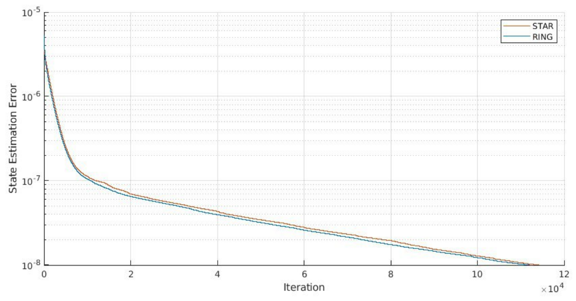

Fig. 15: Star VS Ring Topology state estimation error

Now that we have verified that the algorithm converges under both topologies, we move to the final analysis and we ask whether the consensus reached corresponds to the true target state.
We plotted the state estimation error on a logarithmic scale to better observe the error's evolution over the iterations. As shown in the Figure [15], both topologies converge to zero, indicating that the algorithm is indeed estimating the true state correctly. However, a small but consistent difference can be observed: the star topology exhibits slightly higher estimation error at every iteration compared to the ring topology. Even if both converge, the star network does so more slowly and with less accuracy.

This observation leads to an important conclusion: the choice of network topology significantly impacts the performance of the distributed algorithm. In our experiments, the ring topology proved more efficient, requiring fewer iterations and achieving lower estimation error. This performance of the ring topology can be attributed to its balanced communication structure, which facilitates a symmetric information flow. In contrast, the star topology depends heavily on the central node, which can create bottlenecks and isolate peripheral agents, ultimately impairing both convergence speed and estimation accuracy.

VII. Conclusions

In this project, we faced the problem of secure state estimation and target localization/tracking in Cyber Physical Systems affected by sparse sensor attacks. We explored centralized and distributed algorithms, studying their performance in static and dynamic settings.
In **Task 1**, we compared ISTA and IJAM for system state estimation in the presence of faulty sensors. While both were effective, IJAM generally converged faster, whereas ISTA offered more stable behavior in early iterations.
We also applied ISTA in **Task 2**, to localize a target under sparse sensor attacks using fingerprinting techniques. But as in Task 1, ISTA struggled to recover long term the true target location accurately when sensor attacks were present, even after careful parameter tuning.
Task 3 and **Task 4** we changed directory and extended the problem to dynamic systems. We implemented SSO and DSSO observers to perform secure state estimation and tracking. The

DSSO method, thanks to its deadbeat dynamics, demonstrated faster convergence and lower estimation error, especially in long-term performance, proving to be more effective than SSO. Finally, in **Task 5**, we investigated distributed target localization using the DISTA algorithm. By implementing the consensus-based algorithm on both ring and star network topologies, we demonstrated that convergence is achieved in both cases. However, the ring topology consistently outperformed the star topology, requiring fewer iterations with lower estimation error.

Overall, our results highlight an important consideration: while sparse optimization techniques are powerful tools for mitigating sensor attacks, their robustness strongly depends on the specific context and requirements of the user. In other words, if fast convergence is needed, IJAM is preferable due to its rapid performance. On the other hand, if higher estimation accuracy is the priority, ISTA offers better results despite slower convergence.

Moreover, as discussed throughout the project, the choice of network topology significantly influences the behavior of the distributed algorithm. To achieve optimal performance, a topology that ensures balanced and robust communication is essential, such that information can propagate correctly and efficiently to all agents, especially those located at the network's periphery.