

# Detection and Mitigation of Corrupted Information in Distributed Model Predictive Control Based on Resource Allocation

R. A. Nogueira   R. Bourdais   H. Guéguen

{rafael-accacio.nogueira, romain.bourdais, herve.gueguen} at  
centralesupelec.fr

AUT Department  
IETR — CentraleSupélec

5th International Conference on Control and Fault-Tolerant Systems, 2021



<https://git.io/JEFGW>





- Energy Distribution System
- Traffic management
- Heat distribution
- Water distribution
- ...

# Context



- Energy Distribution System
- Traffic management
- Heat distribution
- Water distribution
- ...





- Energy Distribution System
- Traffic management
- Heat distribution
- Water distribution
- ...



- Energy Distribution System
- Traffic management
- Heat distribution
- Water distribution
- ...



- Geographically distributed
- Coupled by constraints (energy)
- Optimization objectives
  - Energy
  - User satisfaction
  - ...
- Solution → Model Predictive Control



- Geographically distributed
- Coupled by constraints (energy)
- Optimization objectives
  - Energy
  - User satisfaction
  - ...
- Solution → Model Predictive Control



- Geographically distributed
- Coupled by constraints (energy)
- Optimization objectives
  - Energy
  - User satisfaction
  - ...
- Solution → Model Predictive Control





- Geographically distributed
- Coupled by constraints (energy)
- Optimization objectives
  - Energy
  - User satisfaction
  - ...
- Solution  $\rightarrow$  Model Predictive Control

Objective: Find control input sequence that optimizes an objective function

$$\begin{array}{ll} \underset{\mathbf{u}(k:k+N_p-1|k)}{\text{optimize}} & J(\mathbf{x}(k), \mathbf{u}(k)) \\ \text{subject to} & \left. \begin{array}{l} \mathbf{x}(\xi + 1) = f(\mathbf{x}(\xi), \mathbf{u}(\xi)) \\ g_i(\mathbf{x}(\xi), \mathbf{u}(\xi)) \leq 0 \\ h_j(\mathbf{x}(\xi), \mathbf{u}(\xi)) = 0 \end{array} \right\} \begin{array}{l} \forall \xi \in \{1, \dots, N_p\} \\ \forall i \in \{1, \dots, m\} \\ \forall j \in \{1, \dots, p\} \end{array} \end{array}$$

Objective: Find control input sequence that optimizes an objective function

$$\begin{array}{ll} \text{optimize} & J(\mathbf{x}(k), \mathbf{u}(k)) \\ \mathbf{u}(k:k+N_p-1|k) & \\ \text{subject to} & \left. \begin{array}{l} \mathbf{x}(\xi+1) = f(\mathbf{x}(\xi), \mathbf{u}(\xi)) \\ g_i(\mathbf{x}(\xi), \mathbf{u}(\xi)) \leq 0 \\ h_j(\mathbf{x}(\xi), \mathbf{u}(\xi)) = 0 \end{array} \right\} \begin{array}{l} \forall \xi \in \{1, \dots, N_p\} \\ \forall i \in \{1, \dots, m\} \\ \forall j \in \{1, \dots, p\} \end{array} \end{array}$$

Objective: Find control input sequence that optimizes an objective function

$$\begin{array}{ll} \underset{\mathbf{u}(k:k+N_p-1|k)}{\text{optimize}} & J(\mathbf{x}(k), \mathbf{u}(k)) \\ \text{subject to} & \left. \begin{array}{l} \mathbf{x}(\xi + 1) = f(\mathbf{x}(\xi), \mathbf{u}(\xi)) \\ g_i(\mathbf{x}(\xi), \mathbf{u}(\xi)) \leq 0 \\ h_j(\mathbf{x}(\xi), \mathbf{u}(\xi)) = 0 \end{array} \right\} \begin{array}{l} \forall \xi \in \{1, \dots, N_p\} \\ \forall i \in \{1, \dots, m\} \\ \forall j \in \{1, \dots, p\} \end{array} \end{array}$$

Objective: Find control input sequence that optimizes an objective function

$$\begin{array}{ll} \underset{\mathbf{u}(k:k+N_p-1|k)}{\text{optimize}} & J(\mathbf{x}(k), \mathbf{u}(k)) \\ \text{subject to} & \left. \begin{array}{l} \mathbf{x}(\xi+1) = f(\mathbf{x}(\xi), \mathbf{u}(\xi)) \\ g_i(\mathbf{x}(\xi), \mathbf{u}(\xi)) \leq 0 \\ h_j(\mathbf{x}(\xi), \mathbf{u}(\xi)) = 0 \end{array} \right\} \begin{array}{l} \forall \xi \in \{1, \dots, N_p\} \\ \forall i \in \{1, \dots, m\} \\ \forall j \in \{1, \dots, p\} \end{array} \end{array}$$

Objective: Find control input sequence that optimizes an objective function

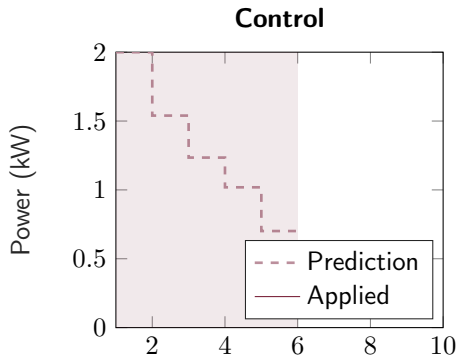
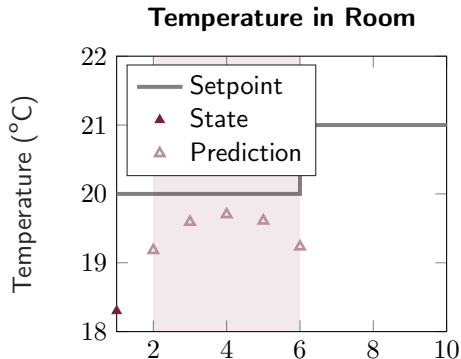
$$\begin{array}{ll} \underset{\mathbf{u}(k:k+N_p-1|k)}{\text{optimize}} & J(\mathbf{x}(k), \mathbf{u}(k)) \\ \text{subject to} & \left. \begin{array}{l} \mathbf{x}(\xi + 1) = f(\mathbf{x}(\xi), \mathbf{u}(\xi)) \\ g_i(\mathbf{x}(\xi), \mathbf{u}(\xi)) \leq 0 \\ h_j(\mathbf{x}(\xi), \mathbf{u}(\xi)) = 0 \end{array} \right\} \begin{array}{l} \forall \xi \in \{1, \dots, N_p\} \\ \forall i \in \{1, \dots, m\} \\ \forall j \in \{1, \dots, p\} \end{array} \end{array}$$

Objective: Find control input sequence that optimizes an objective function

$$\begin{array}{ll} \underset{\mathbf{u}(k:k+N_p-1|k)}{\text{minimize}} & \sum_{j=1}^{N_p} \|\mathbf{v}(k+j|k)\|_Q^2 + \|\mathbf{u}(k+j-1|k)\|_R^2 \\ \text{subject to} & \left. \begin{array}{l} \mathbf{x}(\xi+1) = f(\mathbf{x}(\xi), \mathbf{u}(\xi)) \\ g_i(\mathbf{x}(\xi), \mathbf{u}(\xi)) \leq 0 \\ h_j(\mathbf{x}(\xi), \mathbf{u}(\xi)) = 0 \end{array} \right\} \begin{array}{l} \forall \xi \in \{1, \dots, N_p\} \\ \forall i \in \{1, \dots, m\} \\ \forall j \in \{1, \dots, p\} \end{array} \end{array}$$

# Model Predictive Control

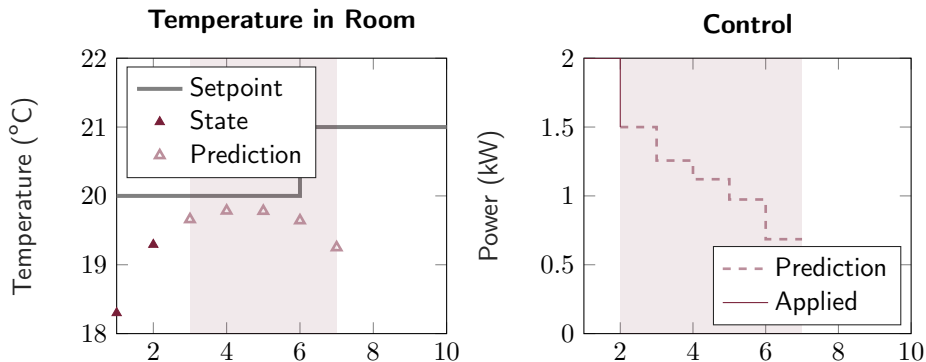
Find optimal control sequence





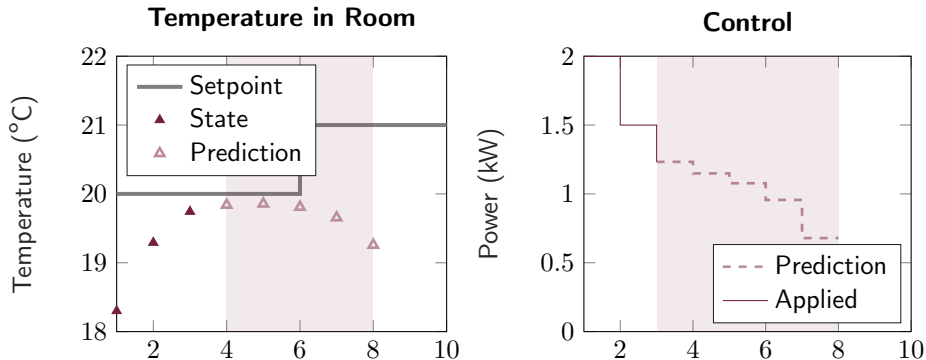
# Model Predictive Control

Find optimal control sequence, apply first element



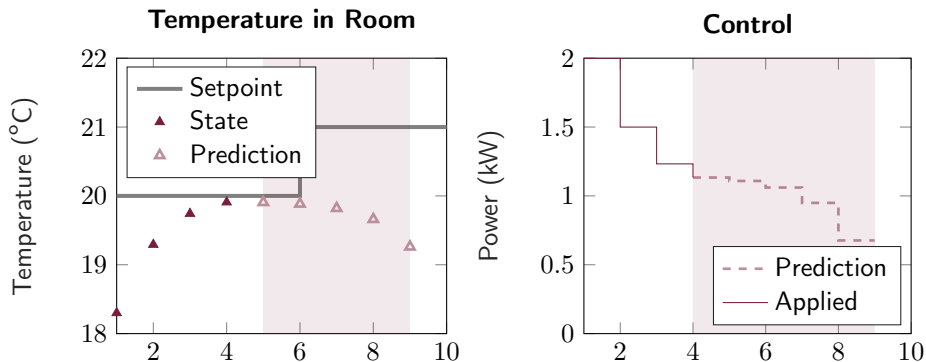
# Model Predictive Control

Find optimal control sequence, apply first element, rinse repeat



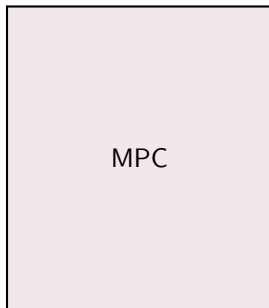
# Model Predictive Control

Find optimal control sequence, apply first element, rinse repeat → Receding Horizon



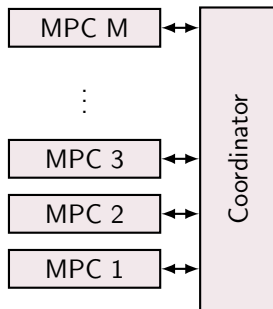
# Distributed Model Predictive Control

- Problem: Complexity depends on  $N_p, m, p$  and sizes of  $x$  and  $u$
- Solution: Divide and Conquer



# Distributed Model Predictive Control

- Problem: Complexity depends on  $N_p, m, p$  and sizes of  $x$  and  $u$
- Solution: Divide and Conquer



# Distributed Model Predictive Control

Quantity Decomposition | Resource Allocation

$$\begin{aligned}
 & \text{minimize}_{\mathbf{u}_i(k:k+N_p-1|k)} \quad \overbrace{\sum_{i=1}^M \sum_{j=1}^{N_p} \|\mathbf{v}_i(k+j|k)\|_{Q_i}^2 + \|\mathbf{u}_i(k+j-1|k)\|_{R_i}^2}^{J_G(k)} \\
 & \text{subject to} \quad \left. \begin{aligned} & \mathbf{x}_i(k+1) = A_i \mathbf{x}_i(k) + B_i \mathbf{u}_i(k) \quad \forall i \in \{1, \dots, M\} \\ & \sum_{i=1}^M \Gamma_i \mathbf{u}_i(k) = \mathbf{u}_{\max} \quad \forall j \in \{1, \dots, N_p\} \end{aligned} \right\} \quad \overbrace{\hspace{10em}}^{J_i(k)}
 \end{aligned}$$



# Distributed Model Predictive Control

Quantity Decomposition | Resource Allocation

$$\left. \begin{aligned} J_i^*(\boldsymbol{\theta}_i(k)) &= \underset{\mathbf{u}_i(k:k+N_p-1|k)}{\text{minimize}} \quad J_i(k) \\ \text{s.t.} \quad \mathbf{x}_i(k+1) &= \mathbf{A}_i \mathbf{x}_i(k) + \mathbf{B}_i \mathbf{u}_i(k) \\ \Gamma_i \mathbf{u}_i(k) &= \boldsymbol{\theta}_i(k) : \boldsymbol{\lambda}_i(k) \end{aligned} \right\} \begin{aligned} &\forall i \in \{1, \dots, M\} \\ &\forall j \in \{1, \dots, N_p\} \end{aligned}$$

$$\begin{aligned} J^* &= \underset{\boldsymbol{\theta}(k:k+N_p-1|k)}{\text{minimize}} \quad \sum_{i=1}^M J_i^*(\boldsymbol{\theta}_i(k)) \\ \text{s.t.} \quad &\sum_{i=1}^M \boldsymbol{\theta}_i(k) = \mathbf{u}_{\max} \end{aligned}$$



# Distributed Model Predictive Control

Quantity Decomposition | Resource Allocation

$$\left. \begin{aligned} J_i^*(\boldsymbol{\theta}_i(k)) &= \underset{\mathbf{u}_i(k:k+N_p-1|k)}{\text{minimize}} \quad J_i(k) \\ \text{s.t.} \quad \mathbf{x}_i(k+1) &= A_i \mathbf{x}_i(k) + B_i \mathbf{u}_i(k) \\ \Gamma_i \mathbf{u}_i(k) &= \boldsymbol{\theta}_i(k) : \boldsymbol{\lambda}_i(k) \end{aligned} \right\} \begin{aligned} &\forall i \in \{1, \dots, M\} \\ &\forall j \in \{1, \dots, N_p\} \end{aligned}$$

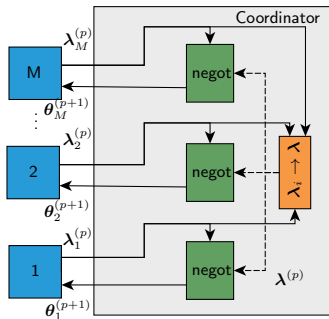
$$\boldsymbol{\theta}_i^{(p+1)} = \boldsymbol{\theta}_i^{(p)} + \rho \left( \boldsymbol{\lambda}_i^*(\boldsymbol{\theta}_i^{(p)}) - \frac{1}{M} \sum_{j=1}^M \boldsymbol{\lambda}_j^*(\boldsymbol{\theta}_j^{(p)}) \right)$$





# Distributed Model Predictive Control

## Quantity Decomposition | Resource Allocation



$$J_i^*(\theta_i(k)) = \underset{\mathbf{u}_i(k:k+N_p-1|k)}{\text{minimize}} J_i(k) \quad \left. \begin{array}{l} \forall i \in \{1, \dots, M\} \\ \text{s.t. } \mathbf{x}_i(k+1) = A_i \mathbf{x}_i(k) + B_i \mathbf{u}_i(k) \\ \Gamma_i \mathbf{u}_i(k) = \theta_i(k) : \lambda_i(k) \end{array} \right\} \forall j \in \{1, \dots, N_p\}$$

$$\theta_i^{(p+1)} = \theta_i^{(p)} + \rho \left( \lambda_i^*(\theta_i^{(p)}) - \frac{1}{M} \sum_{j=1}^M \lambda_j^*(\theta_j^{(p)}) \right)$$

# Quantity Decomposition | Resource Allocation

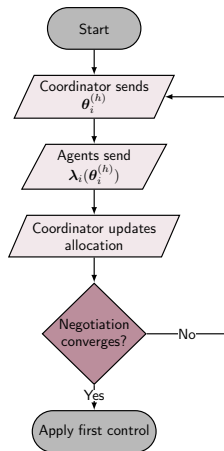


Figure 1: Quantity decomposition based DMPC

# Quantity Decomposition | Resource Allocation

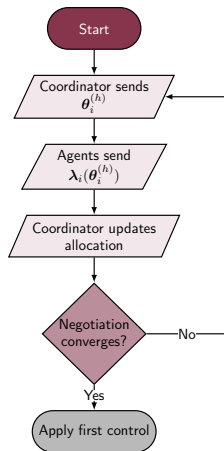


Figure 1: Quantity decomposition based DMPC

# Quantity Decomposition | Resource Allocation

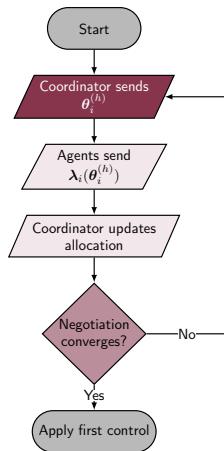


Figure 1: Quantity decomposition based DMPC

# Quantity Decomposition | Resource Allocation

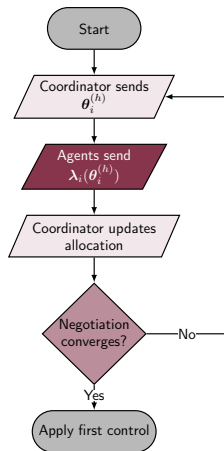


Figure 1: Quantity decomposition based DMPC

# Quantity Decomposition | Resource Allocation

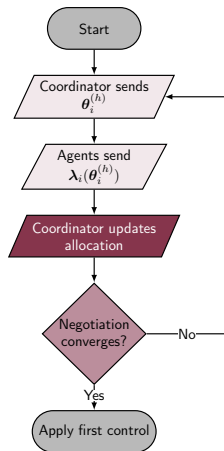


Figure 1: Quantity decomposition based DMPC

# Quantity Decomposition | Resource Allocation

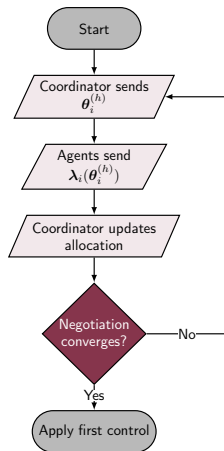


Figure 1: Quantity decomposition based DMPC

# Quantity Decomposition | Resource Allocation

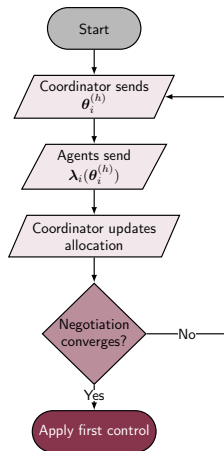


Figure 1: Quantity decomposition based DMPC



What if agents send a non-agreed  $\lambda_i$ ?



- ① Vulnerabilities in distributed MPC based on Resource Allocation
  - Attacks
  - Consequences
- ② Securing the DMPC
  - Analysis of Subproblems
  - Detection Mechanism
  - Mitigation Mechanism
  - Complete Mechanism
- ③ Results

# Outline

- ① Vulnerabilities in distributed MPC based on Resource Allocation
  - Attacks
  - Consequences
- ② Securing the DMPC
  - Analysis of Subproblems
  - Detection Mechanism
  - Mitigation Mechanism
  - Complete Mechanism
- ③ Results



# How can a non-cooperative agent attack?

- $\lambda_i$  is the only interface with coordination
- Non-cooperative agent sends  $\tilde{\lambda}_i = \gamma_i(\lambda_i)$

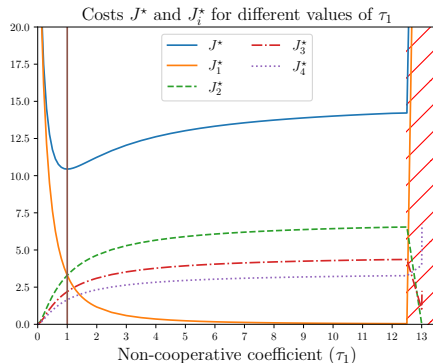


# How can a non-cooperative agent attack?

- $\lambda_i$  is the only interface with coordination
- Non-cooperative agent sends  $\tilde{\lambda}_i = \gamma_i(\lambda_i)$



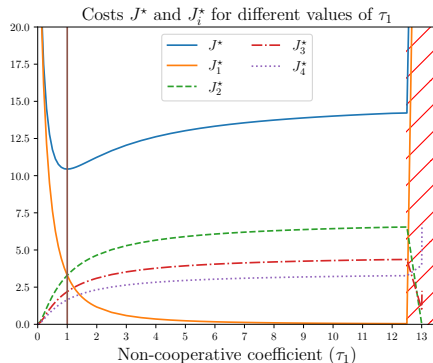
# Example



## 4 distinct agents

- Agent 1 is non-cooperative
- It uses  $\tilde{\lambda}_1 = \gamma_1(\lambda_1) = \tau_1 I \lambda_1$

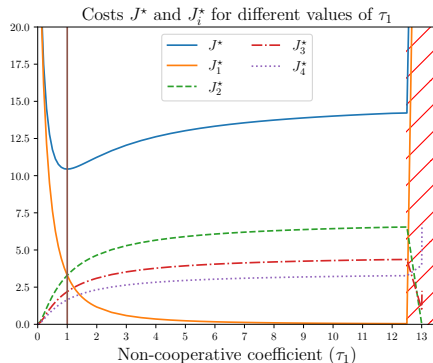
# Example



## 4 distinct agents

- Agent 1 is non-cooperative
- It uses  $\tilde{\lambda}_1 = \gamma_1(\lambda_1) = \tau_1 I \lambda_1$

# Example

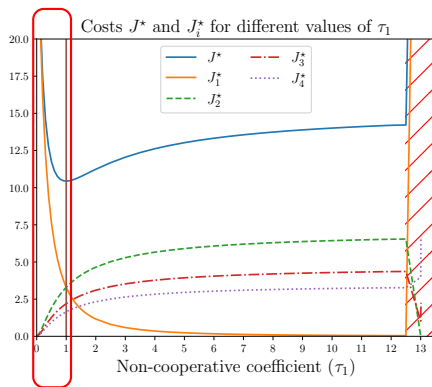


## 4 distinct agents

- Agent 1 is non-cooperative
- It uses  $\tilde{\lambda}_1 = \gamma_1(\lambda_1) = \tau_1 I \lambda_1$



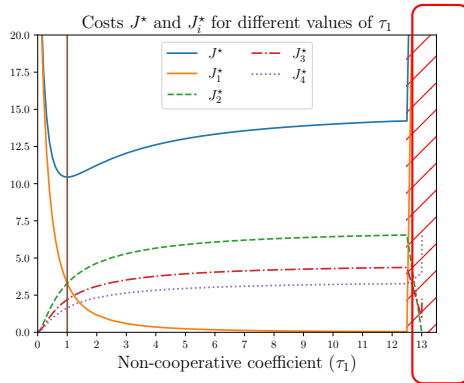
# Example



## 4 distinct agents

- Agent 1 is non-cooperative
- It uses  $\tilde{\lambda}_1 = \gamma_1(\lambda_1) = \tau_1 I \lambda_1$

# Example



## 4 distinct agents

- Agent 1 is non-cooperative
- It uses  $\tilde{\lambda}_1 = \gamma_1(\lambda_1) = \tau_1 I \lambda_1$

# Outline

- ① Vulnerabilities in distributed MPC based on Resource Allocation  
Attacks  
Consequences
- ② Securing the DMPC  
Analysis of Subproblems  
Detection Mechanism  
Mitigation Mechanism  
Complete Mechanism
- ③ Results



# Quadratic Case

$$\begin{aligned}
 & \text{minimize}_{\mathbf{u}_i(k:k+N_p-1|k)} \overbrace{\sum_{j=1}^{N_p} \|\mathbf{v}_i(k+j|k)\|_{Q_i}^2 + \|\mathbf{u}_i(k+j-1|k)\|_{R_i}^2}^{J_i(k)} \\
 & \text{s.t.} \quad \left. \begin{aligned} \mathbf{x}_i(\xi+1) &= A_i \mathbf{x}_i(\xi) + B_i \mathbf{u}_i(\xi) \\ \Gamma_i \mathbf{u}_i(\xi) &= \boldsymbol{\theta}_i(\xi) : \boldsymbol{\lambda}_i(\xi) \end{aligned} \right\} \forall \xi \in \{1, \dots, N_p\}
 \end{aligned}$$

# Quadratic Case

$$\begin{array}{ll} \underset{\mathbf{U}_i(k)}{\text{minimize}} & \overbrace{\frac{1}{2} \mathbf{U}_i(k)^T \mathbf{H}_i \mathbf{U}_i(k) + \mathbf{f}_i(k)^T \mathbf{U}_i(k)}^{J_i(\boldsymbol{\theta}_i)} \\ \text{s.t.} & \boldsymbol{\Theta}_i \mathbf{U}_i(k) = \boldsymbol{\theta}_i : \boldsymbol{\lambda}_i \end{array}$$

# Quadratic Case

$$\begin{array}{ll} \underset{\mathbf{U}_i(k)}{\text{minimize}} & \overbrace{\frac{1}{2} \mathbf{U}_i(k)^T \mathbf{H}_i \mathbf{U}_i(k) + \mathbf{f}_i(k)^T \mathbf{U}_i(k)}^{J_i(\boldsymbol{\theta}_i)} \\ \text{s.t.} & \boldsymbol{\Theta}_i \mathbf{U}_i(k) = \boldsymbol{\theta}_i : \boldsymbol{\lambda}_i \end{array}$$

# Quadratic Case

$$\begin{array}{ll} \underset{U_i(k)}{\text{minimize}} & \overbrace{\frac{1}{2} U_i(k)^T H_i U_i(k) + \textcolor{red}{f}_i(k)^T U_i(k)}^{J_i(\boldsymbol{\theta}_i)} \\ \text{s.t.} & \Theta_i U_i(k) = \boldsymbol{\theta}_i : \lambda_i \end{array}$$

# Quadratic Case

$$\begin{array}{ll} \underset{\mathbf{U}_i(k)}{\text{minimize}} & \overbrace{\frac{1}{2} \mathbf{U}_i(k)^T \mathbf{H}_i \mathbf{U}_i(k) + \mathbf{f}_i(k)^T \mathbf{U}_i(k)}^{J_i(\boldsymbol{\theta}_i)} \\ \text{s.t.} & \boldsymbol{\Theta}_i \mathbf{U}_i(k) = \boldsymbol{\theta}_i : \boldsymbol{\lambda}_i \end{array}$$



# Quadratic Case

$$\begin{array}{ll}
 \underset{\mathbf{U}_i(k)}{\text{minimize}} & \overbrace{\frac{1}{2} \mathbf{U}_i(k)^T \mathbf{H}_i \mathbf{U}_i(k) + \mathbf{f}_i(k)^T \mathbf{U}_i(k)}^{J_i(\boldsymbol{\theta}_i)} \\
 \text{s.t.} & \boldsymbol{\Theta}_i \mathbf{U}_i(k) = \boldsymbol{\theta}_i : \boldsymbol{\lambda}_i
 \end{array}$$

$$\boldsymbol{\lambda}_i = -\mathbf{P}_i \boldsymbol{\theta}_i - \mathbf{s}_i(k)$$

where  $\mathbf{P}_i = (\boldsymbol{\Theta}_i \mathbf{H}_i^{-1} \boldsymbol{\Theta}_i^T)^{-1}$  and  $\mathbf{s}_i(k) = \mathbf{P}_i \boldsymbol{\Theta}_i \mathbf{H}_i^{-1} \mathbf{f}_i(k)$

# Quadratic Case

$$\begin{aligned}
 & \underset{\mathbf{U}_i(k)}{\text{minimize}} && \overbrace{\frac{1}{2} \mathbf{U}_i(k)^T \mathbf{H}_i \mathbf{U}_i(k) + \mathbf{f}_i(k)^T \mathbf{U}_i(k)}^{J_i(\boldsymbol{\theta}_i)} \\
 & \text{s.t.} && \boldsymbol{\Theta}_i \mathbf{U}_i(k) = \boldsymbol{\theta}_i : \boldsymbol{\lambda}_i
 \end{aligned}$$

$$\boldsymbol{\lambda}_i = -\mathbf{P}_i \boldsymbol{\theta}_i - \mathbf{s}_i(k)$$

where  $\mathbf{P}_i = (\boldsymbol{\Theta}_i \mathbf{H}_i^{-1} \boldsymbol{\Theta}_i^T)^{-1}$  and  $\mathbf{s}_i(k) = \mathbf{P}_i \boldsymbol{\Theta}_i \mathbf{H}_i^{-1} \mathbf{f}_i(k)$

# Quadratic Case

$$\begin{aligned}
 & \underset{\mathbf{U}_i(k)}{\text{minimize}} && \overbrace{\frac{1}{2} \mathbf{U}_i(k)^T \mathbf{H}_i \mathbf{U}_i(k) + \mathbf{f}_i(k)^T \mathbf{U}_i(k)}^{J_i(\boldsymbol{\theta}_i)} \\
 & \text{s.t.} && \boldsymbol{\Theta}_i \mathbf{U}_i(k) = \boldsymbol{\theta}_i : \boldsymbol{\lambda}_i
 \end{aligned}$$

$$\boldsymbol{\lambda}_i = -P_i \boldsymbol{\theta}_i - \mathbf{s}_i(k)$$

$$\text{where } P_i = (\boldsymbol{\Theta}_i \mathbf{H}_i^{-1} \boldsymbol{\Theta}_i^T)^{-1} \text{ and } \mathbf{s}_i(k) = P_i \boldsymbol{\Theta}_i \mathbf{H}_i^{-1} \mathbf{f}_i(k)$$

# Detection

## Assumption

*We know nominal  $\bar{P}_i$*

## Assumption

*Attacker chooses  $\tilde{\lambda}_i = \gamma_i(\lambda_i) = T_i(k)\lambda_i \rightarrow -T_i(k)P_i\theta_i - T_i(k)s_i(k)$*

- We can estimate<sup>1</sup>  $\hat{P}_i$  and  $\hat{s}_i(k)$  such as:

$$\tilde{\lambda}_i = \gamma_i(\lambda_i(\theta_i)) = -\hat{P}_i(k)\theta_i - \hat{s}_i(k)$$

- If  $\hat{P}_i(k) \neq \bar{P}_i \rightarrow \text{Attack}$

---

<sup>1</sup>Using Recursive Least Squares

# Detection

## Assumption

*We know nominal  $\bar{P}_i$*

## Assumption

*Attacker chooses  $\tilde{\lambda}_i = \gamma_i(\lambda_i) = T_i(k)\lambda_i \rightarrow -T_i(k)P_i\theta_i - T_i(k)s_i(k)$*

- We can estimate<sup>1</sup>  $\hat{P}_i$  and  $\hat{s}_i(k)$  such as:

$$\tilde{\lambda}_i = \gamma_i(\lambda_i(\theta_i)) = -\hat{P}_i(k)\theta_i - \hat{s}_i(k)$$

- If  $\hat{P}_i(k) \neq \bar{P}_i \rightarrow \text{Attack}$

---

<sup>1</sup>Using Recursive Least Squares

# Detection

## Assumption

*We know nominal  $\bar{P}_i$*

## Assumption

*Attacker chooses  $\tilde{\lambda}_i = \gamma_i(\lambda_i) = T_i(k)\lambda_i \rightarrow -T_i(k)P_i\theta_i - T_i(k)s_i(k)$*

- We can estimate<sup>1</sup>  $\hat{P}_i$  and  $\hat{s}_i(k)$  such as:

$$\tilde{\lambda}_i = \gamma_i(\lambda_i(\theta_i)) = -\hat{P}_i(k)\theta_i - \hat{s}_i(k)$$

- If  $\hat{P}_i(k) \neq \bar{P}_i \rightarrow \text{Attack}$

---

<sup>1</sup>Using Recursive Least Squares

# Detection

## Assumption

*We know nominal  $\bar{P}_i$*

## Assumption

*Attacker chooses  $\tilde{\lambda}_i = \gamma_i(\lambda_i) = T_i(k)\lambda_i \rightarrow -T_i(k)P_i\theta_i - T_i(k)s_i(k)$*

- We can estimate<sup>1</sup>  $\hat{P}_i$  and  $\hat{s}_i(k)$  such as:

$$\tilde{\lambda}_i = \gamma_i(\lambda_i(\theta_i)) = -\hat{P}_i(k)\theta_i - \hat{s}_i(k)$$

- If  $\hat{P}_i(k) \neq \bar{P}_i \rightarrow \text{Attack}$

---

<sup>1</sup>Using Recursive Least Squares

# About Estimation

- We estimate  $\hat{P}_i$  and  $\hat{\mathbf{s}}_i(k)$  simultaneously using Recursive Least Squares
- Problem: Estimation during negotiation fails
  - Consecutive  $\lambda_i^p$  and  $\theta_i^p$  are linearly dependent  $\rightarrow$  low input excitation
- Solution: Send sequence of random values of  $\theta_i$  until estimates converge





# About Estimation

- We estimate  $\hat{P}_i$  and  $\hat{\mathbf{s}}_i(k)$  simultaneously using Recursive Least Squares
- Problem: Estimation during negotiation fails
  - Consecutive  $\lambda_i^p$  and  $\theta_i^p$  are linearly dependent  $\rightarrow$  low input excitation
- Solution: Send sequence of random values of  $\theta_i$  until estimates converge



# About Estimation

- We estimate  $\hat{P}_i$  and  $\hat{\mathbf{s}}_i(k)$  simultaneously using Recursive Least Squares
- Problem: Estimation during negotiation fails
  - Consecutive  $\lambda_i^p$  and  $\theta_i^p$  are linearly dependent  $\rightarrow$  low input excitation
- Solution: Send sequence of random values of  $\theta_i$  until estimates converge



# About Estimation

- We estimate  $\hat{P}_i$  and  $\hat{\mathbf{s}}_i(k)$  simultaneously using Recursive Least Squares
- Problem: Estimation during negotiation fails
  - Consecutive  $\lambda_i^p$  and  $\theta_i^p$  are linearly dependent  $\rightarrow$  low input excitation
- Solution: Send sequence of random values of  $\theta_i$  until estimates converge



# Detection

## In detail

- Error  $E_i(k) = \|\hat{\tilde{P}}_i(k) - \bar{P}_i\|_F$
- Create threshold  $\epsilon_P$
- Indicator  $d_i \in \{0, 1\}$  detects the attack in agent  $i$ .
- $d_i = 1$  if  $E_i(k) > \epsilon_P$ , 0 otherwise



# Detection

## In detail

- Error  $E_i(k) = \|\hat{\tilde{P}}_i(k) - \bar{P}_i\|_F$
- Create threshold  $\epsilon_P$
- Indicator  $d_i \in \{0, 1\}$  detects the attack in agent  $i$ .
- $d_i = 1$  if  $E_i(k) > \epsilon_P$ , 0 otherwise



# Detection

## In detail

- Error  $E_i(k) = \|\hat{\tilde{P}}_i(k) - \bar{P}_i\|_F$
- Create threshold  $\epsilon_P$
- Indicator  $d_i \in \{0, 1\}$  detects the attack in agent  $i$ .
- $d_i = 1$  if  $E_i(k) > \epsilon_P$ , 0 otherwise



# Detection

## In detail

- Error  $E_i(k) = \|\hat{\tilde{P}}_i(k) - \bar{P}_i\|_F$
- Create threshold  $\epsilon_P$
- Indicator  $d_i \in \{0, 1\}$  detects the attack in agent  $i$ .
- $d_i = 1$  if  $E_i(k) > \epsilon_P$ , 0 otherwise



# Mitigation

- Main idea: Reconstruct  $\lambda_i$  and use in negotiation

## Assumption

*We suppose  $\tilde{\lambda}_i = 0$  only if  $\lambda_i = 0$ , which implies  $T_i(k)$  invertible.*

- Estimate the inverse of  $T_i(k)$

$$\widehat{T_i(k)^{-1}} = \bar{P}_i \hat{\tilde{P}}_i(k)^{-1}$$

- Reconstruct  $\lambda_i$

$$\lambda_{i\text{rec}} = \widehat{T_i(k)^{-1}} \tilde{\lambda}_i = -\bar{P}_i \theta_i - \widehat{T_i(k)^{-1}} \hat{\tilde{s}}_i(k)$$





# Mitigation

- Main idea: Reconstruct  $\lambda_i$  and use in negotiation

## Assumption

*We suppose  $\tilde{\lambda}_i = \mathbf{0}$  only if  $\lambda_i = \mathbf{0}$ , which implies  $T_i(k)$  invertible.*

- Estimate the inverse of  $T_i(k)$

$$\widehat{T_i(k)^{-1}} = \bar{P}_i \hat{P}_i(k)^{-1}$$

- Reconstruct  $\lambda_i$

$$\lambda_{i\text{rec}} = \widehat{T_i(k)^{-1}} \tilde{\lambda}_i = -\bar{P}_i \theta_i - \widehat{T_i(k)^{-1}} \hat{s}_i(k)$$



# Mitigation

- Main idea: Reconstruct  $\lambda_i$  and use in negotiation

## Assumption

*We suppose  $\tilde{\lambda}_i = \mathbf{0}$  only if  $\lambda_i = \mathbf{0}$ , which implies  $T_i(k)$  invertible.*

- Estimate the inverse of  $T_i(k)$

$$\widehat{T_i(k)^{-1}} = \bar{P}_i \hat{\tilde{P}}_i(k)^{-1}$$

- Reconstruct  $\lambda_i$

$$\lambda_{i\text{rec}} = \widehat{T_i(k)^{-1}} \tilde{\lambda}_i = -\bar{P}_i \theta_i - \widehat{T_i(k)^{-1}} \hat{\tilde{s}}_i(k)$$



# Mitigation

- Main idea: Reconstruct  $\lambda_i$  and use in negotiation

## Assumption

*We suppose  $\tilde{\lambda}_i = 0$  only if  $\lambda_i = 0$ , which implies  $T_i(k)$  invertible.*

- Estimate the inverse of  $T_i(k)$

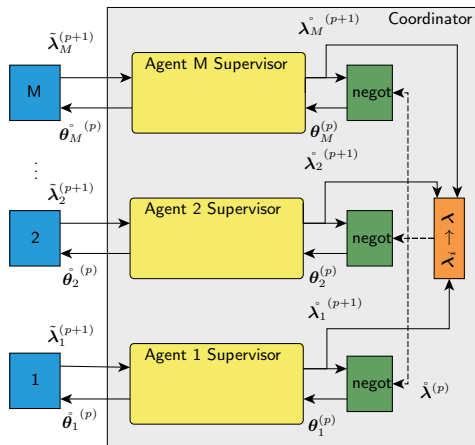
$$\widehat{T_i(k)^{-1}} = \bar{P}_i \hat{\tilde{P}}_i(k)^{-1}$$

- Reconstruct  $\lambda_i$

$$\lambda_{i\text{rec}} = \widehat{T_i(k)^{-1}} \tilde{\lambda}_i = -\bar{P}_i \theta_i - \widehat{T_i(k)^{-1}} \hat{\tilde{s}}_i(k)$$



# Complete Mechanism



Two phases:

- 1 Detect which agents are non-cooperative
- 2 Reconstruct  $\lambda_i$  and use in negotiation

# Secure DMPC

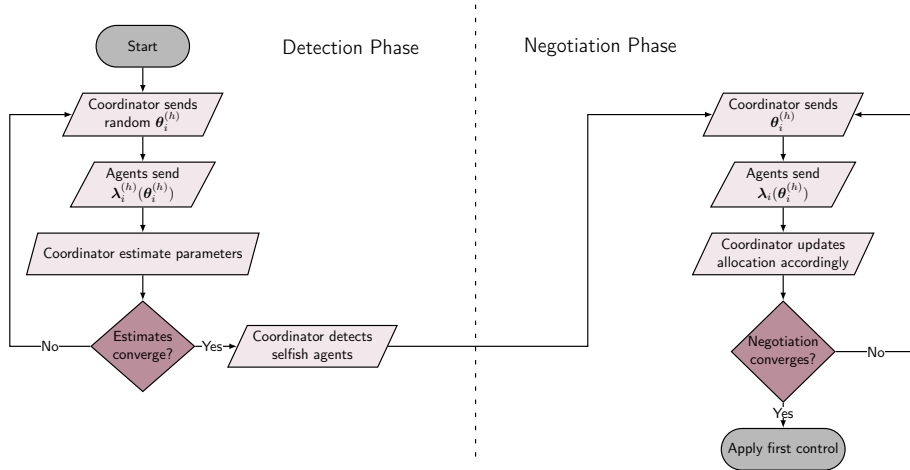


Figure 2: Secure DMPC

# Secure DMPC

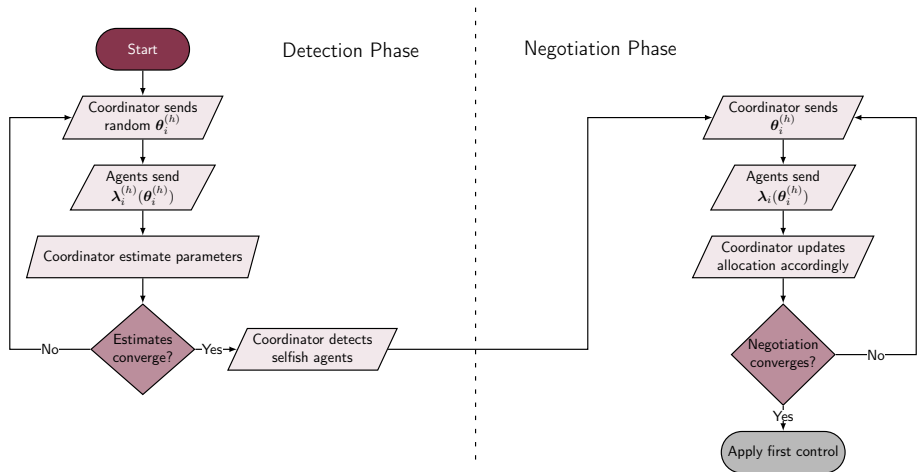


Figure 2: Secure DMPC

# Secure DMPC

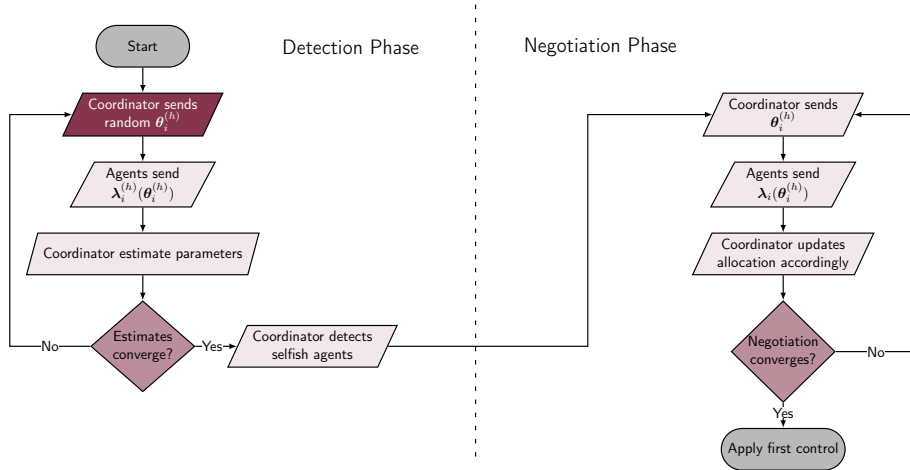


Figure 2: Secure DMPC

# Secure DMPC

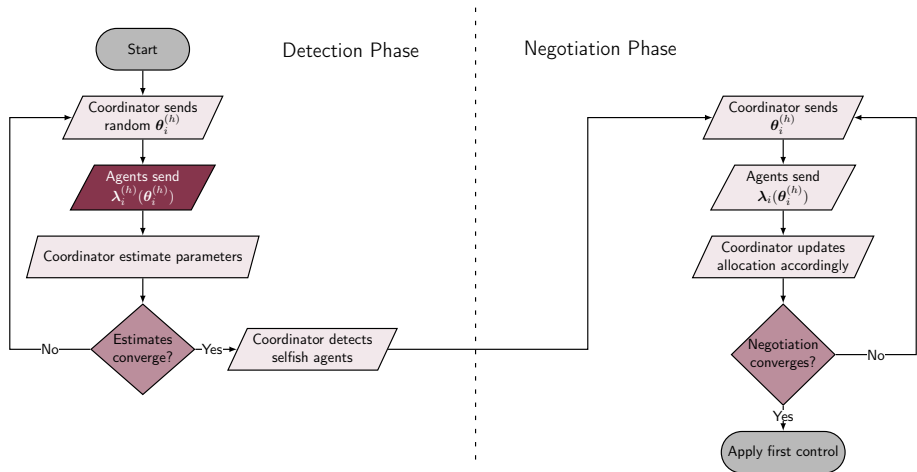


Figure 2: Secure DMPC



## Secure DMPC

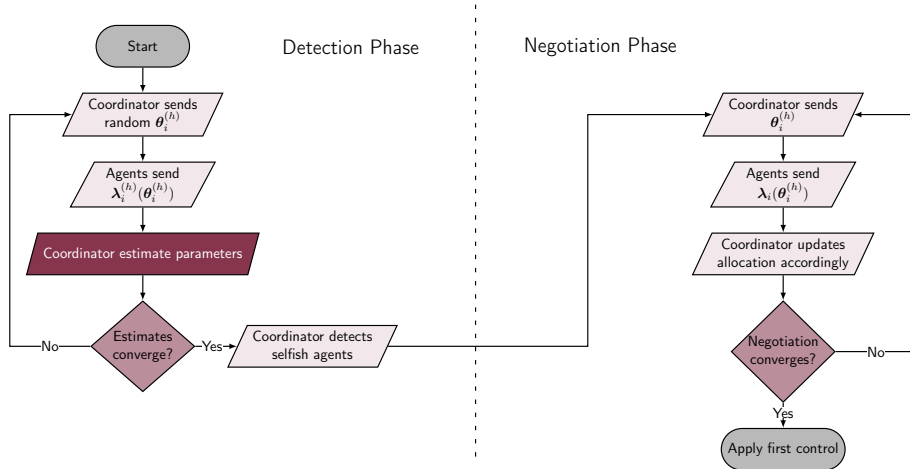


Figure 2: Secure DMPC

# Secure DMPC

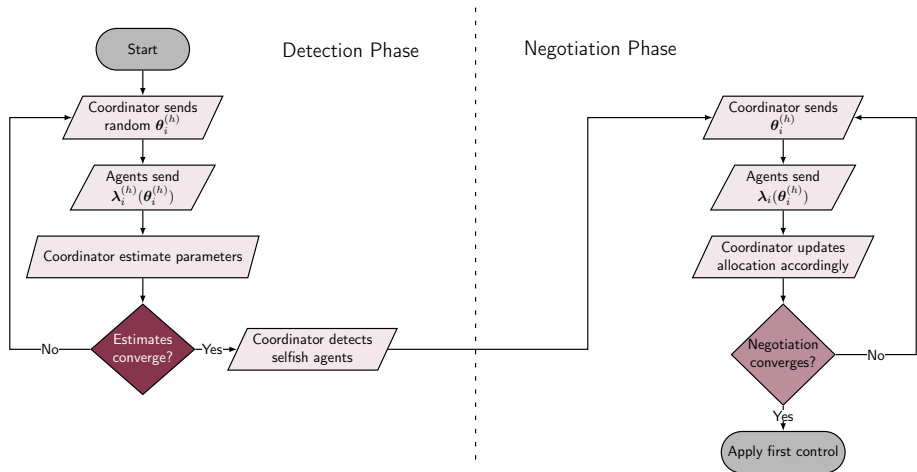


Figure 2: Secure DMPC

# Secure DMPC

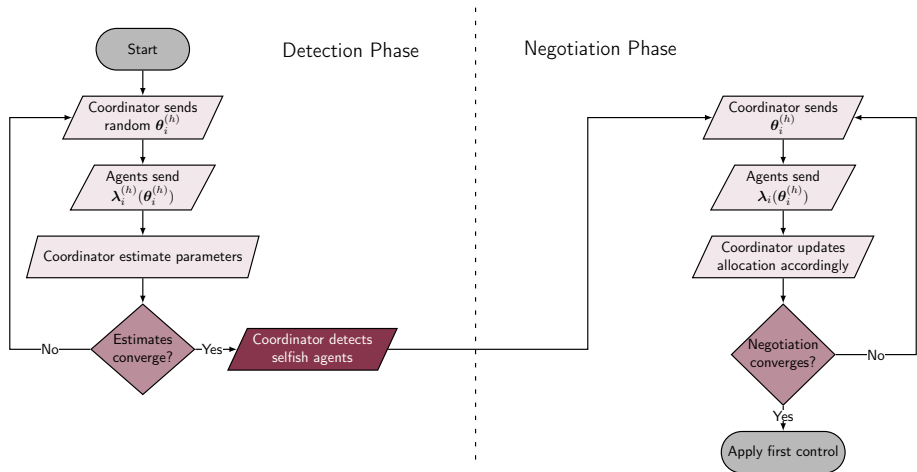


Figure 2: Secure DMPC

# Secure DMPC

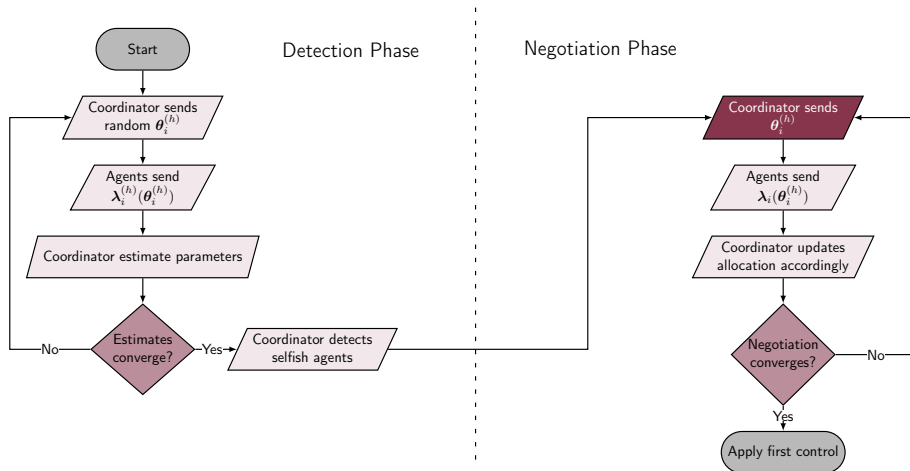


Figure 2: Secure DMPC

# Secure DMPC

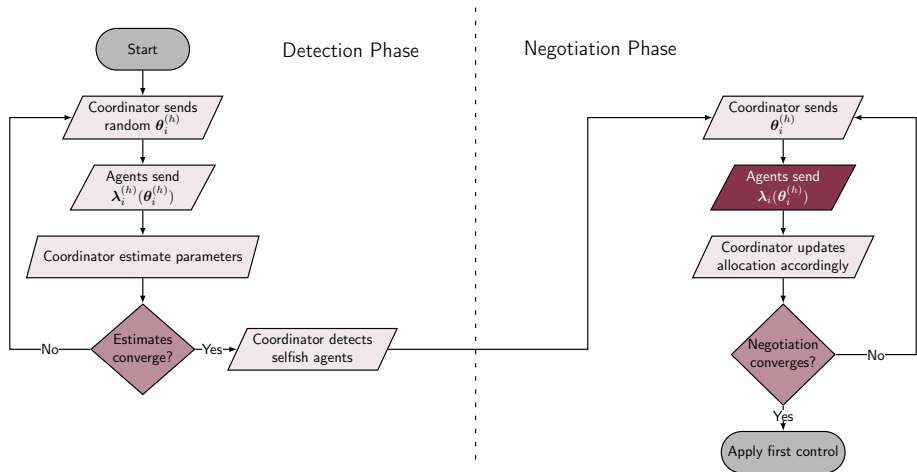


Figure 2: Secure DMPC

# Secure DMPC

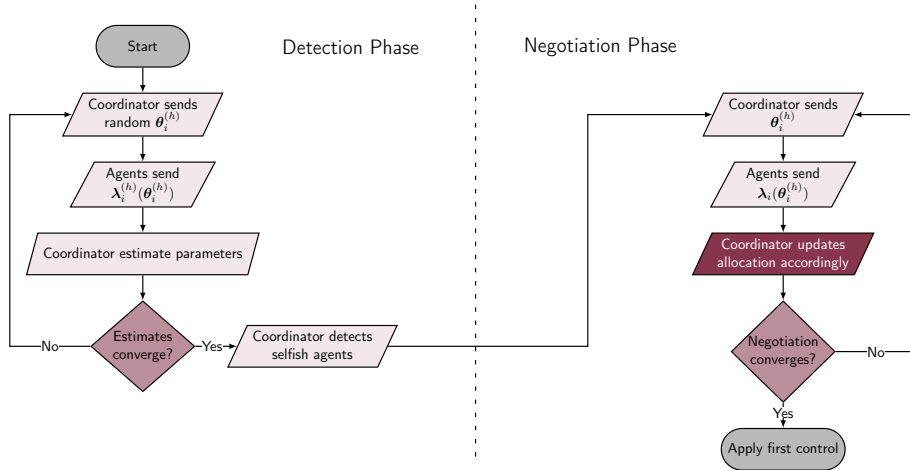


Figure 2: Secure DMPC

## Secure DMPC

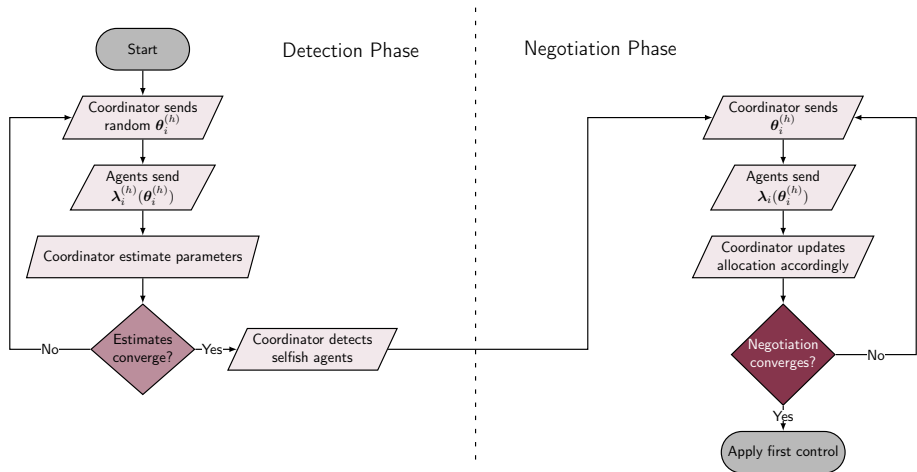


Figure 2: Secure DMPC

# Secure DMPC

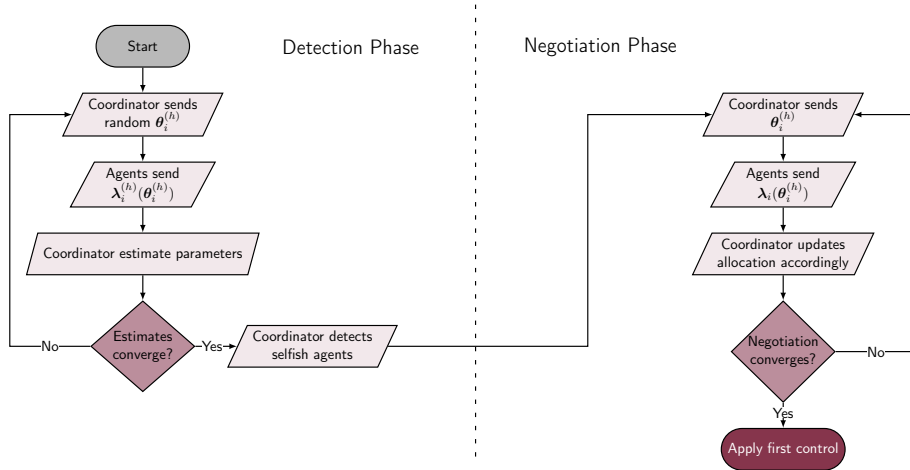


Figure 2: Secure DMPC



# Outline

- ① Vulnerabilities in distributed MPC based on Resource Allocation
  - Attacks
  - Consequences
- ② Securing the DMPC
  - Analysis of Subproblems
  - Detection Mechanism
  - Mitigation Mechanism
  - Complete Mechanism
- ③ Results



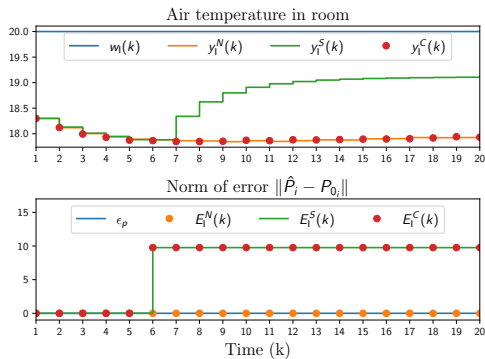
# Example

## Temperature Control of 4 Distinct Rooms Under Power Scarcity

- 4 distinct rooms modeled using 3R-2C
- Initial temperature under  $20^{\circ}\text{C}$
- Not enough power to achieve setpoint  $\left(\sum_{i=1}^4 u_i(k) \leq 4\text{kW}\right)$
- Simulated for a period of 5h
- ZOH  $T_s = 0.25\text{h}$

## Results

## Temporal



N Nominal

S Selfish behavior

C selfish behavior with Correction

# Results

Table 1: Comparison of costs  $J_i^N$  and  $J_G^N$

Agent	Nominal	Selfish	Selfish + correction
I	103	64	104
II	73	91	73
III	100	123	101
IV	132	154	131
Global	408	442	409

# Summary

- ① Resource allocation based DMPC is vulnerable to attacks.
  - ② Sub-problems' structure has time invariant parameters.
  - ③ Attacks can be detected using these parameters.
  - ④ Effects can be mitigated.
- Outlook
    - Inequality Constraints yield Hybrid behavior
    - Non-linear attack model



# Summary

- ① Resource allocation based DMPC is vulnerable to attacks.
  - ② Sub-problems' structure has time invariant parameters.
  - ③ Attacks can be detected using these parameters.
  - ④ Effects can be mitigated.
- Outlook
    - Inequality Constraints yield Hybrid behavior
    - Non-linear attack model



# Summary

- ① Resource allocation based DMPC is vulnerable to attacks.
  - ② Sub-problems' structure has time invariant parameters.
  - ③ Attacks can be detected using these parameters.
  - ④ Effects can be mitigated.
- Outlook
    - Inequality Constraints yield Hybrid behavior
    - Non-linear attack model



# Summary

- ① Resource allocation based DMPC is vulnerable to attacks.
  - ② Sub-problems' structure has time invariant parameters.
  - ③ Attacks can be detected using these parameters.
  - ④ Effects can be mitigated.
- Outlook
    - Inequality Constraints yield Hybrid behavior
    - Non-linear attack model





# Summary

- ① Resource allocation based DMPC is vulnerable to attacks.
  - ② Sub-problems' structure has time invariant parameters.
  - ③ Attacks can be detected using these parameters.
  - ④ Effects can be mitigated.
- Outlook
    - Inequality Constraints yield Hybrid behavior
    - Non-linear attack model



# Summary

- ① Resource allocation based DMPC is vulnerable to attacks.
  - ② Sub-problems' structure has time invariant parameters.
  - ③ Attacks can be detected using these parameters.
  - ④ Effects can be mitigated.
- Outlook
    - Inequality Constraints yield Hybrid behavior
    - Non-linear attack model



# Summary

- ① Resource allocation based DMPC is vulnerable to attacks.
  - ② Sub-problems' structure has time invariant parameters.
  - ③ Attacks can be detected using these parameters.
  - ④ Effects can be mitigated.
- Outlook
    - Inequality Constraints yield Hybrid behavior
    - Non-linear attack model





# Summary

- ① Resource allocation based DMPC is vulnerable to attacks.
  - ② Sub-problems' structure has time invariant parameters.
  - ③ Attacks can be detected using these parameters.
  - ④ Effects can be mitigated.
- Outlook
    - Inequality Constraints yield Hybrid behavior
    - Non-linear attack model



## For Further Reading I

-  J. M. Maestre, R. R. Negenborn *et al.*  
*Distributed Model Predictive Control made easy.*  
Springer, 2014, vol. 69.
-  P. Velarde, J. M. Maestre, H. Ishii, and R. R. Negenborn,  
“Scenario-based defense mechanism for distributed model predictive control,”  
*2017 IEEE 56th Annual Conference on Decision and Control (CDC)*. IEEE,  
Dec 2017, pp. 6171–6176.



Thank you!

Repository

<https://github.com/Accacio/SysTol-21>



Contact

[rafael-accacio.nogueira@centralesupelec.fr](mailto:rafael-accacio.nogueira@centralesupelec.fr)

