FORCES
FOUNDATIONS OF RESILIENT CYBER-PHYSICAL SYSTEMS

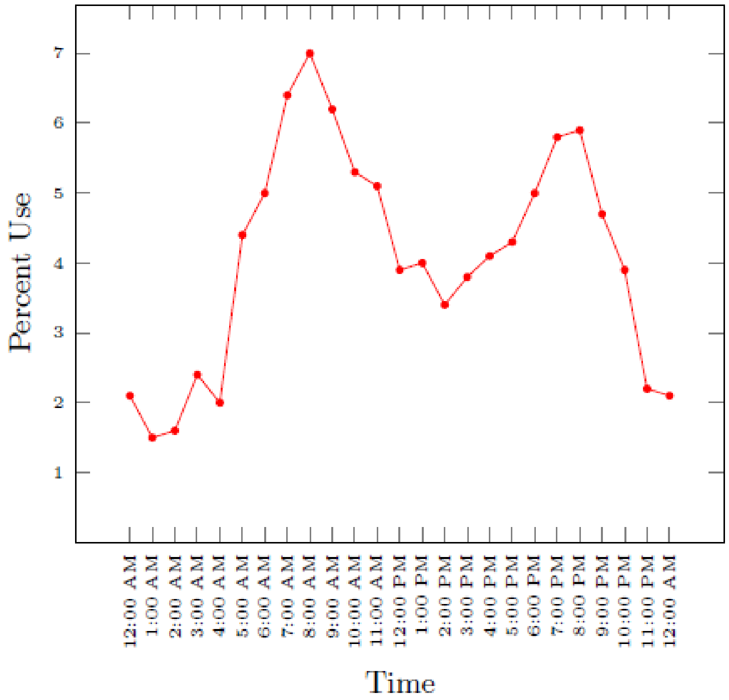
A Game-Theoretic Approach for Selecting Optimal Thresholds for Anomaly Detection in Dynamical Environments

Amin Ghafouri, Aron Laszka, Waseem Abbas,
Yevgeniy Vorobeychik, and Xenofon Koutsoukos
Institute for Software Integrated Systems
Vanderbilt University

Berkeley UNIVERSITY OF CALIFORNIA
Massachusetts Institute of Technology
NSF
MICHIGAN
VANDERBILT UNIVERSITY

Example: Contamination Attack in Water Distribution System

- Attackers may **contaminate** water in a water distribution system.
- The expected **damage** of contamination is **high**, when **water demand** is **high**.
- **Objective**: Design a **detection framework** that detects attacks and **minimizes damage**.



Daily demand pattern

Anomaly Detector

1. **Predictor**: Given **previous water quality** measurements (e.g., pH, chlorine), predicts **current measurements**
2. **Statistical Test**: **Compares** prediction and observation
   - Compute residual $r_k = ||\text{prediction} - \text{observation}||$, then:

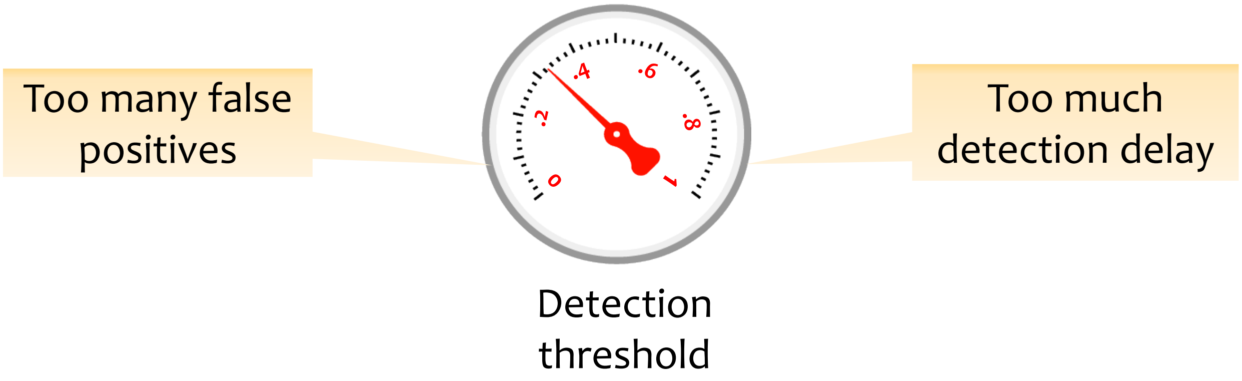
$$S(r_k) \underset{\text{Normal}}{\overset{\text{Anomaly}}{\gtrless}} \eta_k$$

Statistics (e.g., CUSUM, EWMA, etc.)

Threshold

Trade-off Between Detection Delay and False Positives

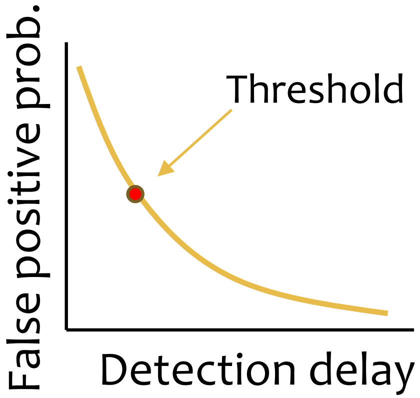
- **Detector metrics**: Detection delay, False positive probability
- **Trade-off** between detection delay and FP that depends on **threshold**



Too many false positives

Too much detection delay

Detection threshold



Problem: Find **thresholds** that **minimize losses** due to **detection delay** and **false positives** considering worst-case contamination attacks.

Stackelberg Game for Optimal Thresholds

Strategic Choices:

1) Defender:
Selects **time-dependent threshold** for the detector

2) Attacker:
Selects a **start time** and an **attack type**

Defender's Loss:

Loss due to False Positive

Loss due to Threshold change

$$\mathcal{L}(\boldsymbol{\eta}, k_a, \lambda) = \sum_{k=1}^{T} C_f \cdot FP(\eta_k) + \sum_{k=k_a}^{\sigma(\boldsymbol{\eta}, k_a, \lambda)} \mathcal{D}(k, \lambda) + N \cdot C_d$$

Loss due to Attack

Attacker's Payoff:

$$\mathcal{P}(\boldsymbol{\eta}, k_a, \lambda) = \sum_{k=k_a}^{\sigma(\boldsymbol{\eta}, k_a, \lambda)} \mathcal{D}(k, \lambda)$$

Optimal Threshold Problem

- **Optimal Threshold Problem**: Minimizes the defender's loss given that the attacker plays a best-response.

$$\boldsymbol{\eta}^* \in \underset{\substack{\boldsymbol{\eta}, \\ (k_a,\lambda) \in \mathrm{bestResponses}(\boldsymbol{\eta})}}{\arg\min} \mathcal{L}(\boldsymbol{\eta}, k_a, \lambda),$$

where $\mathrm{bestResponses}(\boldsymbol{\eta})$ is the set of best-response attacks against a threshold and

$$\mathcal{L}(\boldsymbol{\eta}, k_a, \lambda) = \sum_{k=1}^{T} C_f \cdot FP(\eta_k) + \sum_{k=k_a}^{\sigma(\boldsymbol{\eta}, k_a, \lambda)} \mathcal{D}(k, \lambda) + N \cdot C_d$$

Algorithm for Computing Optimal Threshold

- The algorithm consists of
  - 1) A **dynamic-programming** algorithm for finding **minimum-cost thresholds** subject to the constraint that the damage caused by a best-response attack is at most a given **damage bound.**
  - 2) An **exhaustive search** that finds an optimal damage bound and thereby **optimal thresholds.**

- **Theorem:** Algorithm computes **optimal** thresholds that minimize the defender's loss.
  - Proof) See paper.
  - Running time: $\mathcal{O}(T^2 \cdot |\Delta|^{|\Lambda|+2} \cdot |\Lambda|^2 \cdot |E|)$

A. Ghafouri, Aron Laszka, Waseem Abbas, Yevgeniy Vorobeychik, and Xenofon Koutsoukos, "A Game-Theoretic Approach for Selecting Optimal Thresholds for Attack Detection in Dynamical Environments." To be submitted to Automatica.

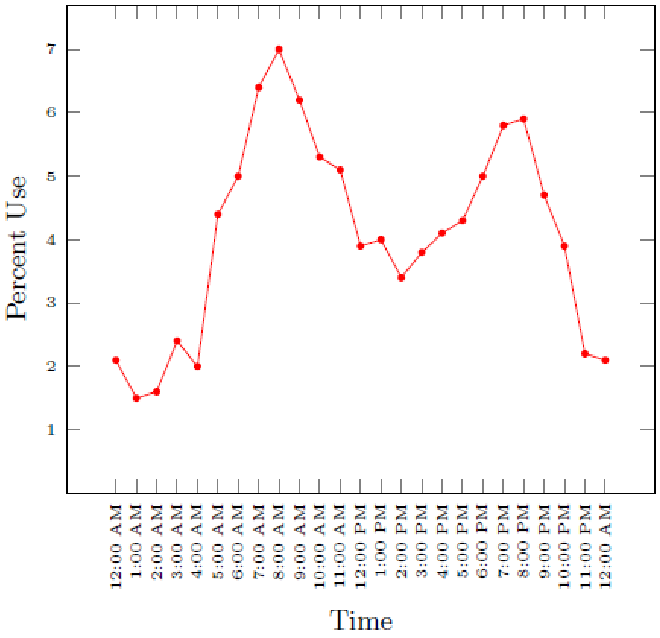
Case Study: Water Contamination

- 6 weeks of **water quality** measurements collected by a utility in the US[1].
- Attacker **contaminates** water with toxic chemical types $\lambda \in \{1.5, 2, 2.5, 3, 4, 5\}$.

$$x_{\text{contaminated}} = \mathcal{F}(x_k, \lambda, \sigma_k, \mu_k)$$

- **Damage** is a function of chemical type and demand:

$$\mathcal{D}(k, \lambda) = (\lambda - 1) \cdot d(k)$$

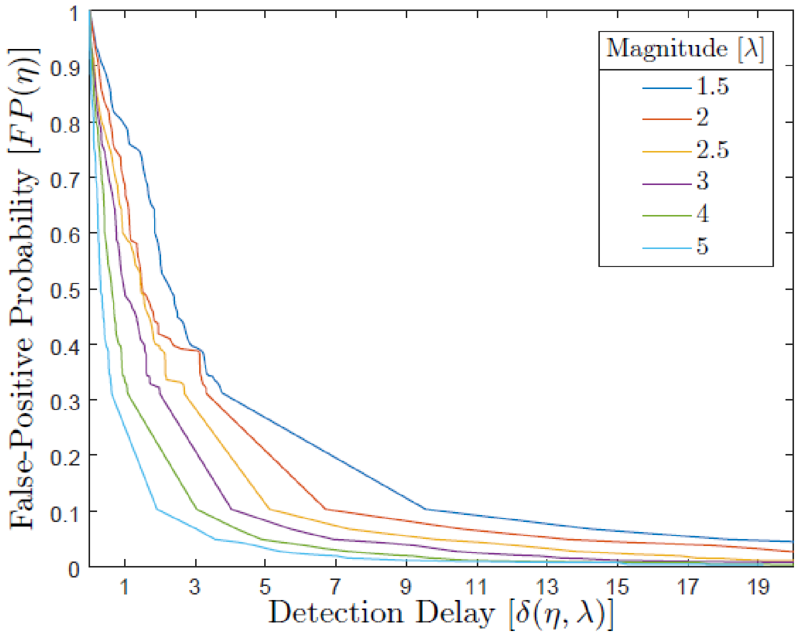


[1] Links, Hot. "CANARY: A Water Quality Event Detection Tool."

Anomaly Detector

1. **Predictor**
   - Feed-Forward Neural Network
   - *Input*: **Lagged** measurement of **target** variable and **current measurements** of other water quality parameters

2. **Statistical Test (CUSUM)**
   - 1,000 simulations for each threshold
   - **Trade-off curve for Chlorine**
   - Detection error decreases as attack magnitude increases

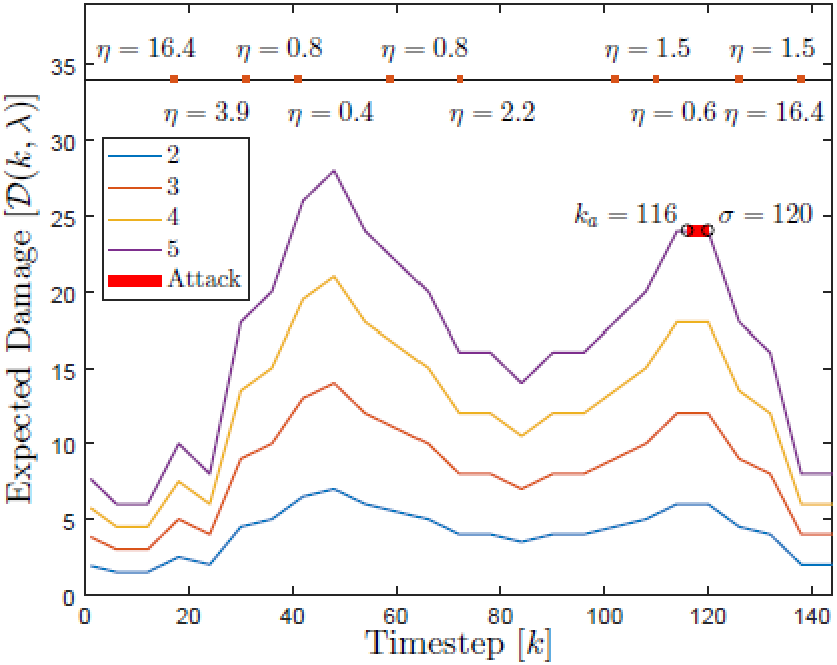


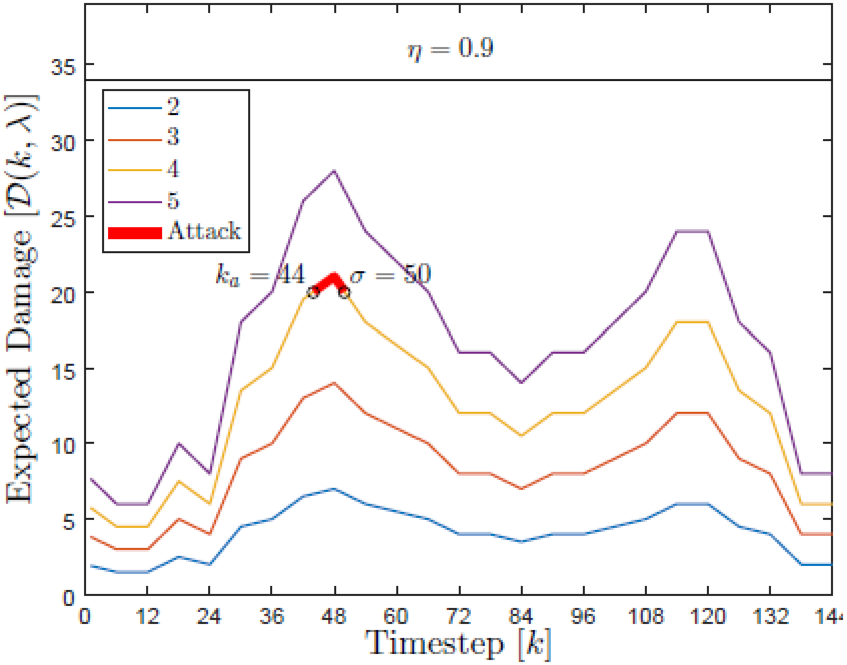
Results

Threshold **decreases** during **critical** periods and **increases** during **non-critical** periods

Fixed Threshold

$$L^* = 222.45$$
$$P^* = 144$$



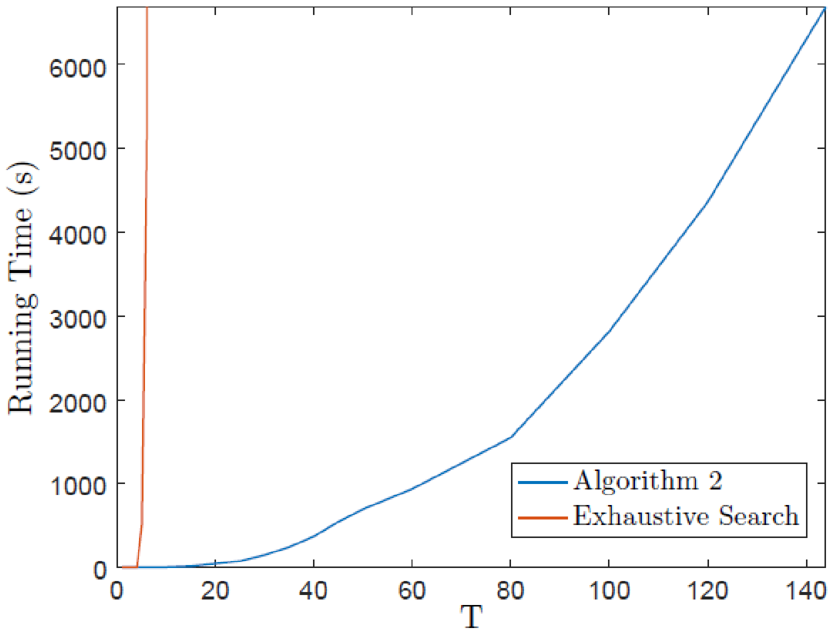


Simulation & Running Time

- Theoretical model vs. Simulation of realistic operation (42 days)
  - Expected: L = 187.72, P = 120
  - Relative error between theoretical loss and actual loss: **4.26%**

	Loss	Payoff	Delay	Number of FPs
Mean	195.83	110.29	3.71	5.60
STD	4.66	8.87	0.31	0.25
MSE	87.04	127.99	0.12	0.43

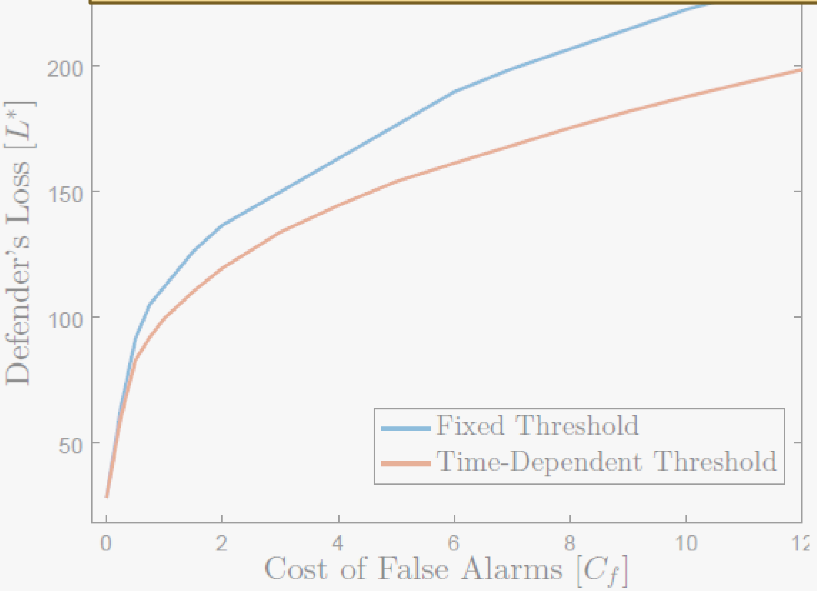
- Running Time of time-dependent threshold algorithm vs. exhaustive search

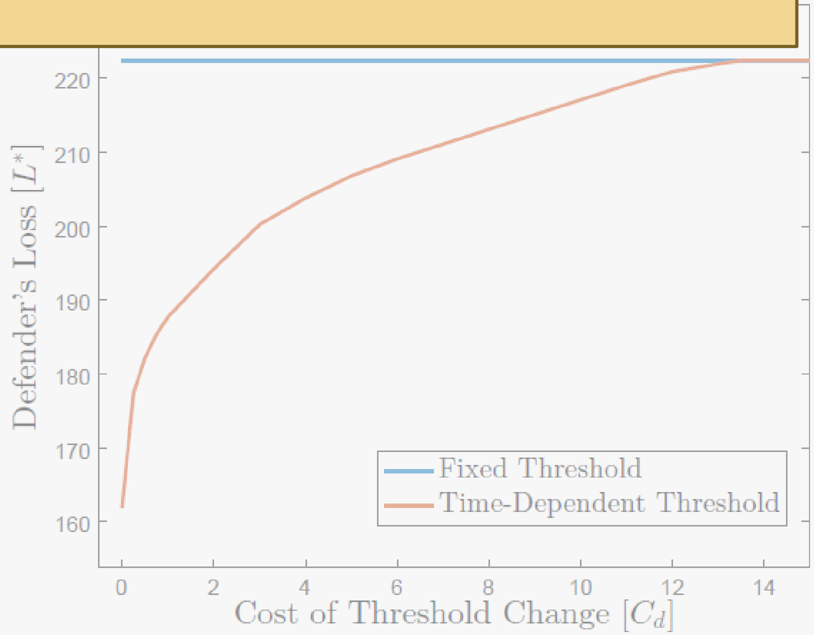


Sensitivity Analysis

- Time-dependent threshold **reduces** the loss by **up to 30%.**
- Improvement compared to fixed threshold:

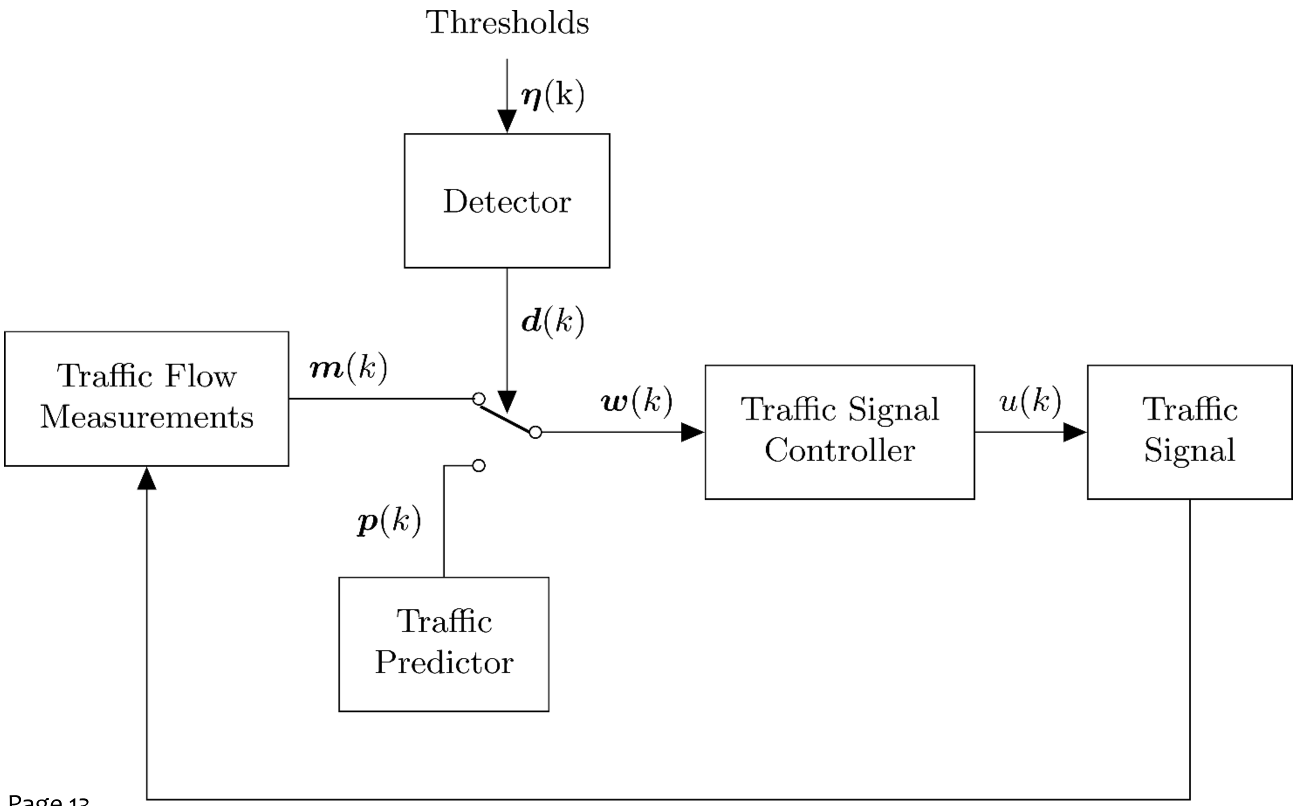
Conclusion: By taking into account **time-varying** aspects of CPS**,** we can reduce **losses** due to attacks and false alarms by **~30%.**





Ongoing Work:
Application-Aware Detection In Traffic Networks

- Optimal detection can be applied to for example, **real-time control** of **traffic signals**:

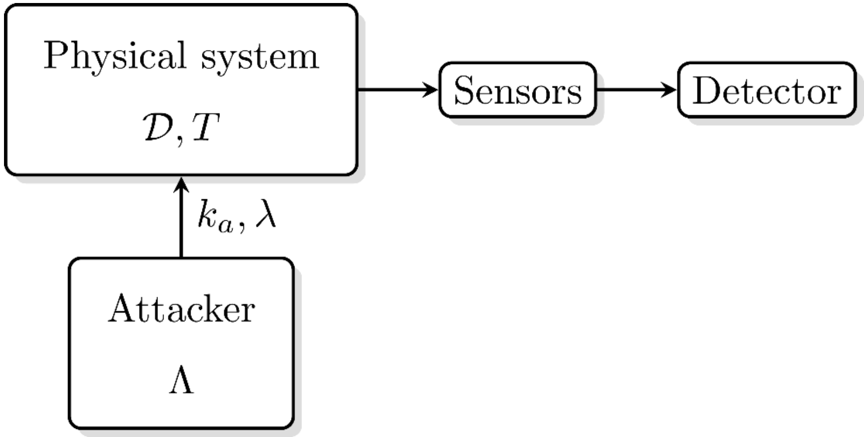


Thank you for your attention!

Questions?

System Model

- CPS with a finite **time horizon** of interest $\{1, \ldots, T\}$
- Detector is deployed in CPS.
- Adversaries may perform an attack of **type** $\lambda \in \Lambda$ (e.g., type of toxic chemical).
- Attack starts at **time** k_a
- **Damage Function**: *Represents the **expected damage** $\mathcal{D}(k, \lambda)$ incurred by the system from an **attack** of type λ at time k.*



ALGORITHM 1: MinimumCostThresholds(P)

```
$\forall \boldsymbol{m} \in \Delta^{|\Lambda|},\ \eta \in E:$ Cost$(T+1, \boldsymbol{m}, \eta) \leftarrow 0$
for $n = T, \dots, 1$ do
    forall $\boldsymbol{m} \in \Delta^{|\Lambda|}$ do
        forall $\eta_{prev} \in E$ do
            if $\bigvee_{\lambda \in \Lambda} \left(\sum_{k=n-m_\lambda}^{n} \mathcal{D}(k, \lambda) > P\right)$ then
                Cost$(n, \boldsymbol{m}, \eta_{\text{prev}}) \leftarrow \infty$
            else
                forall $\eta \in E$ do
                    if $\eta_{prev} = \eta \vee n = 1$ then
                        $S(n, \boldsymbol{m}, \eta_{\text{prev}}, \eta) \leftarrow$ Cost$(n+1,$
                            $\langle \min\{\delta(\eta, \lambda), m_\lambda + 1\} \rangle_{\lambda \in \Lambda}, \eta) + C_f \cdot FP(\eta)$
                    else
                        $S(n, \boldsymbol{m}, \eta_{\text{prev}}, \eta) \leftarrow$ Cost$(n+1,$
                            $\langle \min\{\delta(\eta, \lambda), m_\lambda + 1\} \rangle_{\lambda \in \Lambda}, \eta) + C_f \cdot FP(\eta) + C_d$
                    end
                end
                $\eta^*(n, \boldsymbol{m}, \eta_{prev}) \leftarrow \arg\min_\eta S(n, \boldsymbol{m}, \eta_{prev}, \eta)$
                Cost$(n, \boldsymbol{m}, \eta_{prev}) \leftarrow \min_\eta S(n, \boldsymbol{m}, \eta_{prev}, \eta)$
            end
        end
    end
end
$\boldsymbol{m} \leftarrow \langle 0, \dots, 0 \rangle,\ \eta_0^* \leftarrow$ arbitrary
forall $n = 1, \dots T$ do
    $\eta_n^* \leftarrow \eta^*(n, \boldsymbol{m}, \eta_{n-1}^*)$
    $\boldsymbol{m} \leftarrow \langle \min\{\delta(\eta_n^*, \lambda), m_\lambda + 1\} \rangle_{\lambda \in \Lambda}$
end
return (Cost$(1, \langle 0, \dots, 0 \rangle,$ arbitrary$), \boldsymbol{\eta}^*)$
```

ALGORITHM 2: OptimalThresholds

```
SearchSpace $\leftarrow \left\{ \sum_{k=k_a}^{k_a+\delta} \mathcal{D}(k, \lambda) \;\middle|\; \exists\, k_a \in \{1, \dots, T-1\},\ \delta \in \Delta,\ \lambda \in \Lambda \right\}$
forall $P \in$ SearchSpace do
    $(TC(P), \boldsymbol{\eta}^*(P)) \leftarrow$ MinimumCostThresholds$(P)$
end
$P^* \leftarrow \arg\min_{P \,\in\, \textit{SearchSpace}} TC(P)$
return $\boldsymbol{\eta}^*(P^*)$
```