ACM CCS 2021
November 15-19, 2021 / Online Conference

# Indistinguishability Prevents Scheduler Side Channels in Real-Time Systems

**Chien-Ying (CY) Chen**, Debopam Sanyal and Sibin Mohan

ILLINOIS

Oregon State University

This work is supported by the National Science Foundation (NSF) under grant SaTC-1718952.

# Outline

- Background
  - Real-Time System Security
- Related Work
- $\epsilon$-Scheduler
- Evaluation
  - Simulation-Based Evaluation
  - Application-Based Evaluation
- Summary

# Modern Real-Time Systems

UAV

Ground Delivery Robots

Delivery Drones

Surgical Systems

Real-Time Trading

Self-Driving Cars

Time-Critical Manufacturing Systems

# Why Attack Real-Time Systems?

- Properties of Applications
  - Well-Defined Functionalities
  - Safety-Critical Services
  - High Intellectual/Financial Motivations

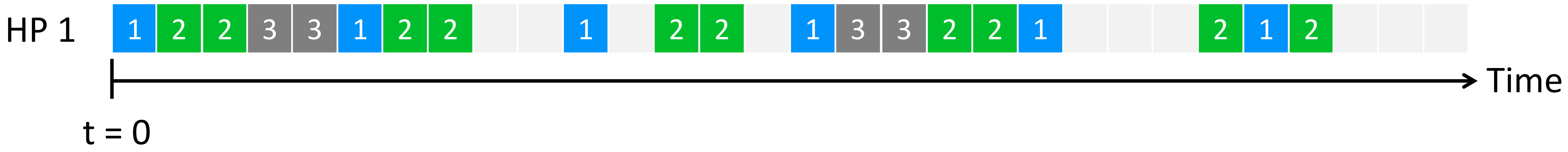
- Properties of Real-Time Systems
  - Time Constraints (Deadlines)
  - Repeated Jobs (Periodic/Sporadic Tasks)
  - Determinism (Worst Case Execution/Response Time Analysis)

**Behavior is highly predictable in RTS!**

# Real-Time Schedules

| | Period | WCET |
|---|---|---|
| 1 | 5 | 1 |
| 2 | 6 | 2 |
| 3 | 15 | 2 |

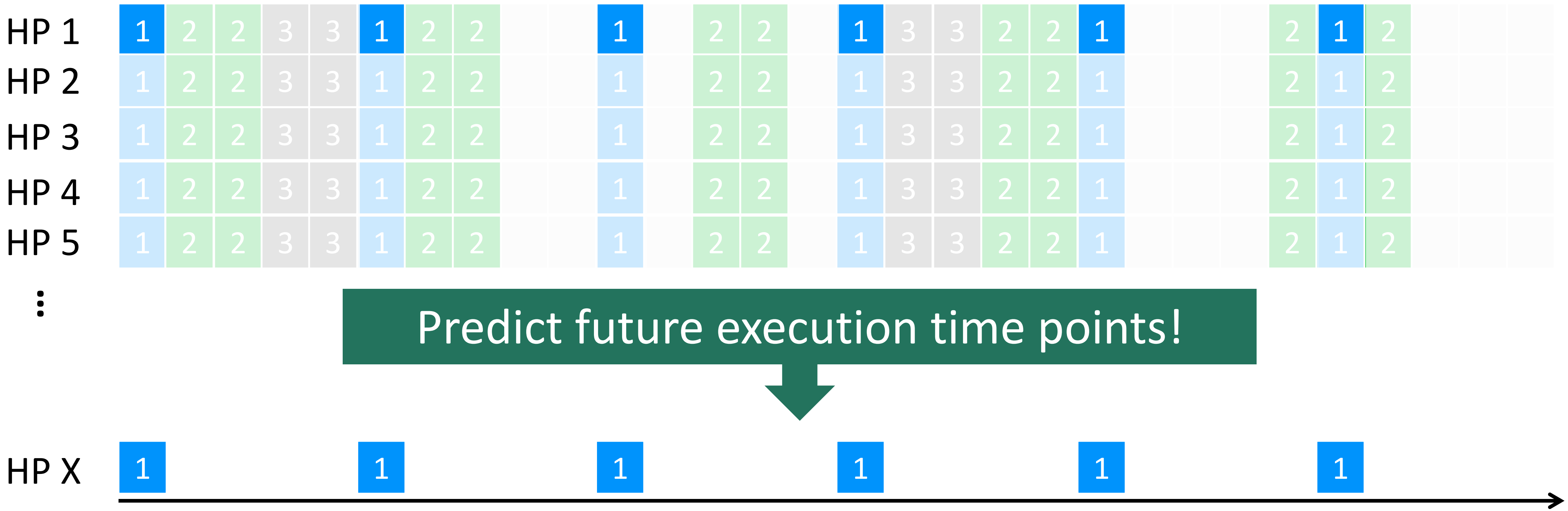
Hyper-Period = LCM(5, 6, 15) = 30



# Real-Time Schedules

| | Period | WCET |
|---|---|---|
| 1 | 5 | 1 |
| 2 | 6 | 2 |
| 3 | 15 | 2 |

Hyper-Period = LCM(5, 6, 15) = 30



| | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| HP 1 | 1 | 2 | 2 | 3 | 3 | 1 | 2 | 2 | | | 1 | | 2 | 2 | | 1 | 3 | 3 | 2 | 2 | 1 | | | | 2 | 1 | 2 | | | |
| HP 2 | 1 | 2 | 2 | 3 | 3 | 1 | 2 | 2 | | | 1 | | 2 | 2 | | 1 | 3 | 3 | 2 | 2 | 1 | | | | 2 | 1 | 2 | | | |
| HP 3 | 1 | 2 | 2 | 3 | 3 | 1 | 2 | 2 | | | 1 | | 2 | 2 | | 1 | 3 | 3 | 2 | 2 | 1 | | | | 2 | 1 | 2 | | | |
| HP 4 | 1 | 2 | 2 | 3 | 3 | 1 | 2 | 2 | | | 1 | | 2 | 2 | | 1 | 3 | 3 | 2 | 2 | 1 | | | | 2 | 1 | 2 | | | |
| HP 5 | 1 | 2 | 2 | 3 | 3 | 1 | 2 | 2 | | | 1 | | 2 | 2 | | 1 | 3 | 3 | 2 | 2 | 1 | | | | 2 | 1 | 2 | | | |
| ⋮ | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| HP X | 1 | | | | | 1 | | | | | 1 | | | | | 1 | | | | | 1 | | | | | 1 | | | | |

Predict future execution time points!

# State of the Art

- Side-Channels
  - Memory/Cache Access Time [1], Branch Prediction [2]
  - Power Consumption Traces [3]
  - Electromagnetic (EM) Emanations [4]
  - Temperature [5]
  - The ScheduLeak Attack Algorithms [6]

[1] Osvik, Dag Arne et al. "Cache attacks and countermeasures: the case of AES." Cryptographers’ track at the RSA conference, 2006.
[2] Kocher, Paul, et al. "Spectre attacks: Exploiting speculative execution." 2019 IEEE Symposium on Security and Privacy (SP). IEEE, 2019.
[3] Jiang, Ke, et al. "Robustness analysis of real-time scheduling against differential power analysis attacks." IEEE Computer Society Annual Symposium on VLSI, 2014
[4] Agrawal, Dakshi, et al. "The EM side—channel (s)." International Workshop on Cryptographic Hardware and Embedded Systems. Springer, Berlin, Heidelberg, 2002.
[5] Bar-El, Hagai, et al. "The sorcerer's apprentice guide to fault attacks." Proceedings of the IEEE 94.2, 2006.
[6] Chen, Chien-Ying, et al. "A Novel Side-Channel in Real-Time Schedulers." 2019 IEEE Real-Time and Embedded Technology and Applications Symposium. IEEE, 2019.

# State of the Art (cont.)

- Defense Strategies in Real-Time Systems
  - Security Tasks Integration [1]
  - Simplex-Based Intrusion Detection Systems [2]
  - Restart-Based Mechanisms [3]
  - Resource Isolation [4]

We focus on defensive techniques in the scheduler

[1] Hasan, Monowar, et al. "Exploring opportunistic execution for integrating security into legacy hard real-time systems." IEEE, RTSS, 2016.
[2] Yoon, Man-Ki, et al. "SecureCore: A multicore-based intrusion detection architecture for real-time embedded systems." 19th IEEE, RTAS, 2013.
[3] Abdi, Fardin, et al. "Preserving Physical Safety Under Cyber Attacks." IEEE Internet of Things Journal, 2018.
[4] Pellizzoni, Rodolfo, et al. "A generalized model for preventing information leakage in hard real-time systems." 21st IEEE, RTAS, 2015.

# State of the Art (cont.)

- Data/Information Protection Techniques
  - Randomization [1] and Resource Isolation [2]
  - Differential Privacy [3]
  - Distributed System Node Privacy [4]
  - Information Hiding [5]

We focus on the system level core properties (e.g. task parameters)

[1] Yoon, Man-Ki, et al. "Taskshuffler: A schedule randomization protocol for obfuscation against timing inference attacks in real-time systems." 20th IEEE, RTAS, 2016.
[2] Pellizzoni, Rodolfo, et al. "A generalized model for preventing information leakage in hard real-time systems." 21st IEEE, RTAS, 2015.
[3] Dwork, Cynthia, and Aaron Roth. "The algorithmic foundations of differential privacy." Foundations and Trends in Theoretical Computer Science 9.3-4 (2014): 211-407.
[4] Z. Huang, et al., “On the cost of differential privacy in distributed control systems,” 3rd HCNS , 2014.
[5] Klara Nahrstedt, Lintian Qiao, "Non-Invertible Watermarking Methods for MPEG Video and Audio", ACM Multimedia (Security Workshop), 1998.

# $\epsilon$-Scheduler

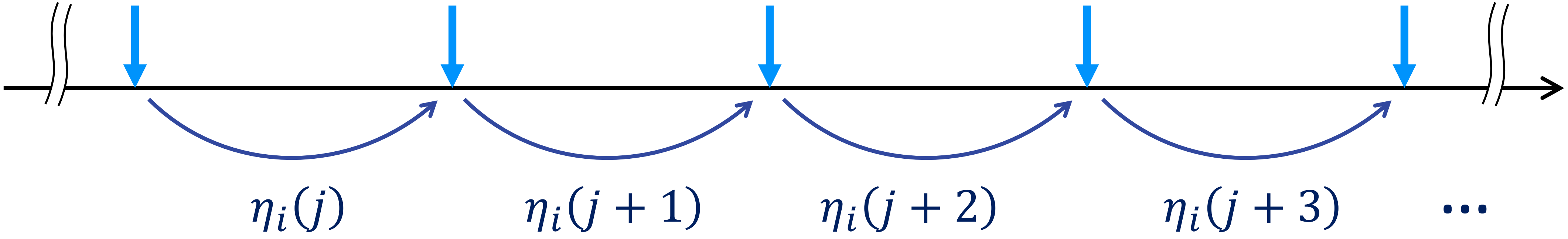
A real-time scheduler that diversifies task schedule by enabling **schedule indistinguishability**

# What $\epsilon$-Scheduler Achieves?

**High Level Goals**

- *Diversify task schedule*
- *Offer analyzable protection*

# Problem Formulation

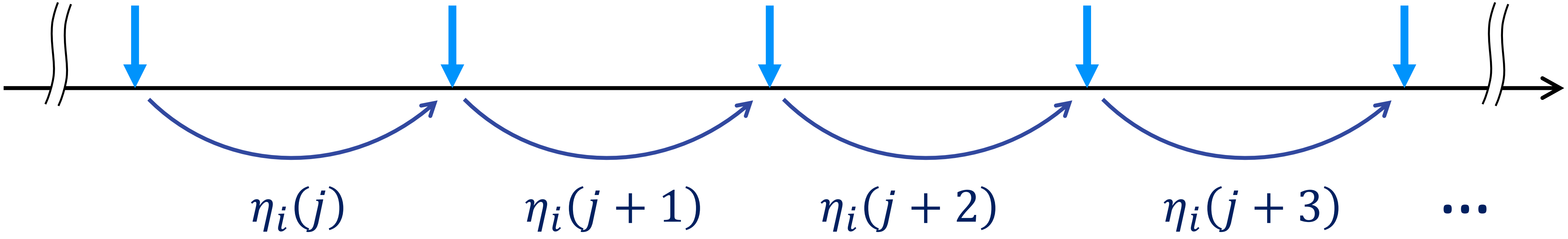


- Task Inter-Arrival Time Function: $\eta_i: \mathbb{N} \longrightarrow T_i$

$$\eta_i(j) = T_{i,j}$$

the inter-arrival time of the task at the j-th instance

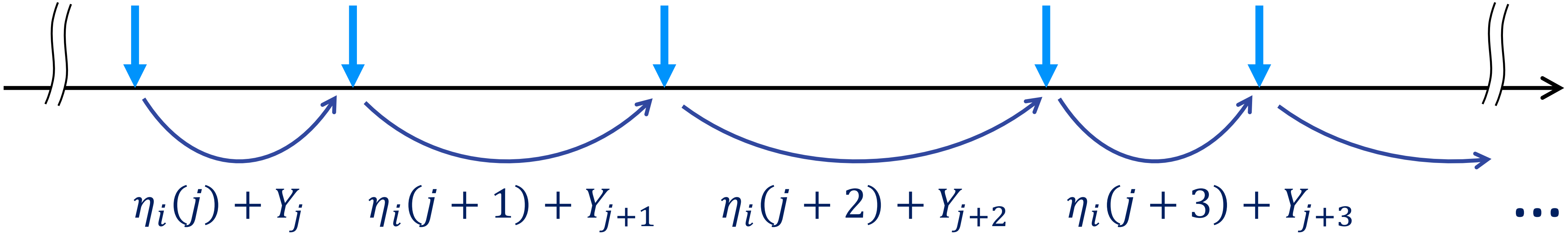
# Schedule Diversification Strategy



- Inter-arrival time randomized mechanism $\mathcal{R}(\tau_i, j)$ :

$$\mathcal{R}(\tau_i, j) = \underbrace{\eta_i(j)}_{\text{task inter-arrival time function}} + \underbrace{Y}_{\text{random noise}}$$

# Schedule Diversification Strategy



- Inter-arrival time randomized mechanism $\mathcal{R}(\tau_i, j)$ :

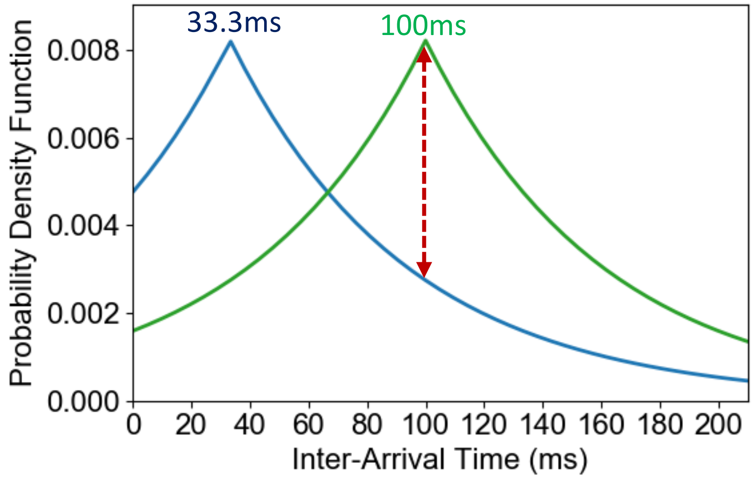
$$\mathcal{R}(\tau_i, j) = \underbrace{\eta_i(j)}_{\text{task inter-arrival time function}} + \underbrace{Y}_{\text{random noise}}$$

*How to design an effective $\mathcal{R}(\cdot)$ for schedule diversification and analyzable protection?*

# Schedule Indistinguishability

- The difficulty of distinguishing a job’s arrival from another
- ”Schedule Indistinguishability” is formally defined as:

$$\underbrace{\frac{\Pr[\,\mathcal{R}(\tau, j) \in S\,]}{\Pr[\,\mathcal{R}(\tau', j') \in S\,]}}_{\text{ratio of any two inter-arrival time distributions}}$$

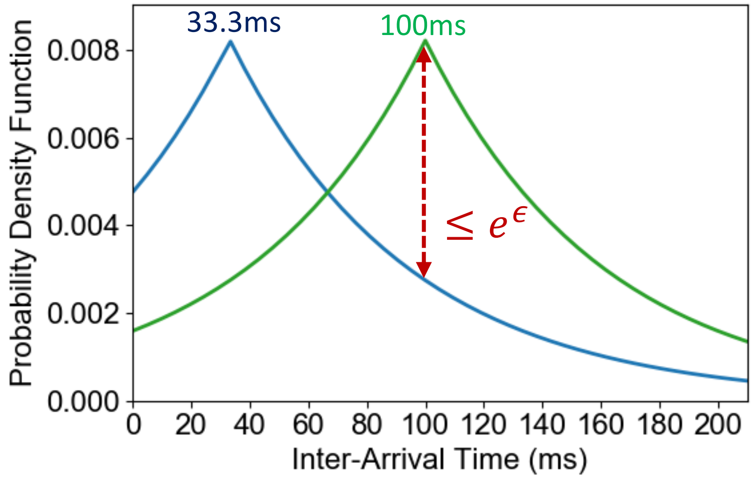


*We want the ratio to be small*
*so that it is hard to distinguish two inter-arrival times*

# Schedule Indistinguishability

- The difficulty of distinguishing a job's arrival from another
- "Schedule Indistinguishability" is formally defined as:

$$\underbrace{\frac{\Pr[\, \mathcal{R}(\tau, j) \in S \,]}{\Pr[\, \mathcal{R}(\tau', j') \in S \,]}}_{\substack{\text{ratio of any two} \\ \text{inter-arrival time distributions}}} \le \ e^{\epsilon}$$

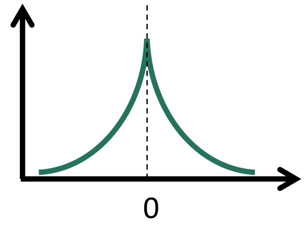


- If a mechanism $\mathcal{R}(\tau, j)$ can yield a ratio $\le \ e^{\epsilon}$, then a **$\epsilon$-indistinguishability** is achieved.
- The $\epsilon$ value becomes an indistinguishability parameter.

# $\epsilon$-Indistinguishable Randomized Mechanism

- Laplace Distribution-Based Noise

$$\mathcal{R}(\tau_i, j) = \boxed{Lap(\cdot)}$$



- Factors to consider for determining noise scale in RTS:

Noise Sensitivity

Duration of Protection

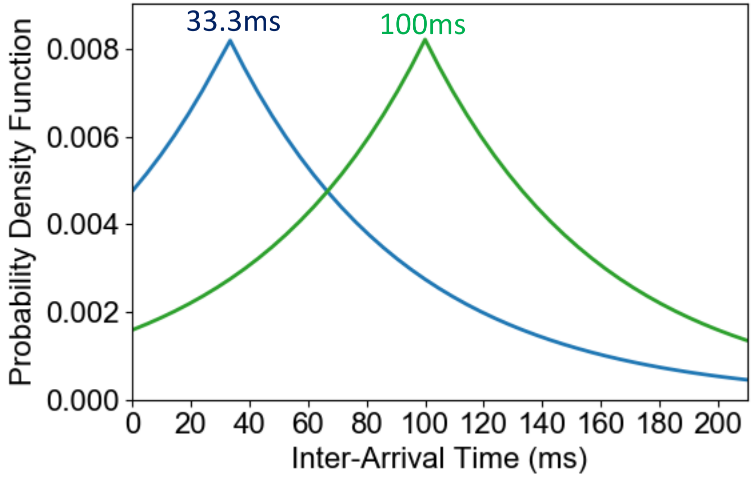
Inter-arrival Time Bound

# Noise Sensitivity

- Inter-Arrival Time Sensitivity

$$\Delta\eta_i =: \max_{\substack{\tau,\tau'\in\Gamma\\ j,j'\in\mathbb{N}}} |\eta_\tau(j) - \eta_{\tau'}(j')|$$

distance between any two possible inter-arrival times

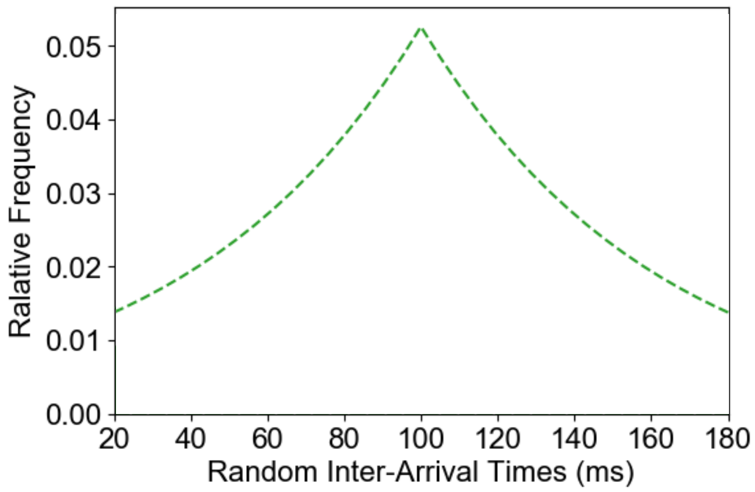


How large the noise should be to make any two inter-arrival times indistinguishable?

*The sensitivity $\Delta\eta_i$ determines the base distribution scale*

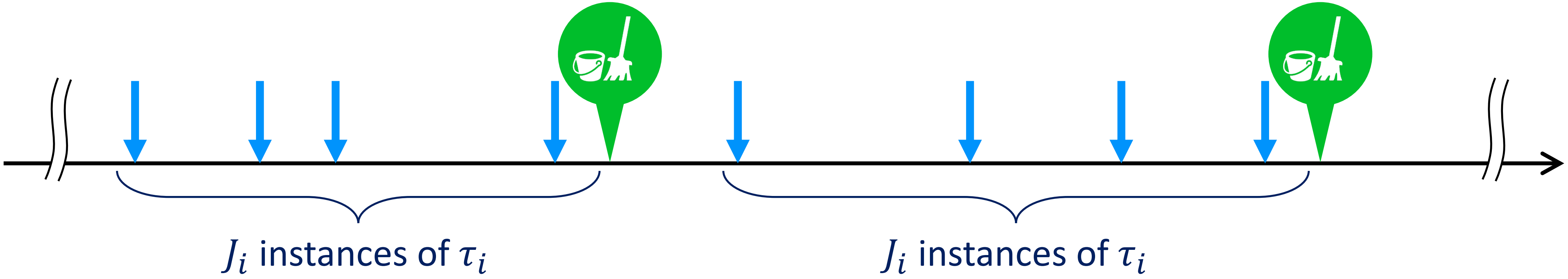
# Duration of Protection ⏱

- Attackers getting sufficient samples may reconstruct the noise distribution
- Adjust the noise scale to ensure $\epsilon$-indistinguishability up to $J_i$ instances



# Duration of Protection ⏱ (cont.)

- Ensure $\epsilon$-indistinguishability within $J_i$ instances
- Integrate with other defense techniques that enforce security checks
  - ➢ Security task integration [1]
  - ➢ Restart-based mechanism [2]

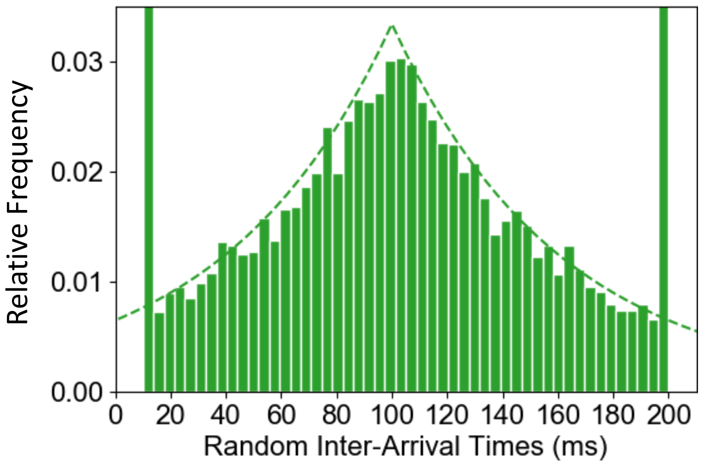


[1] Hasan, Monowar, et al. "Exploring opportunistic execution for integrating security into legacy hard real-time systems." IEEE, RTSS, 2016.
[2] Abdi, Fardin, et al. "Preserving Physical Safety Under Cyber Attacks." IEEE Internet of Things Journal, 2018.

# Inter-Arrival Time Bound →| |←

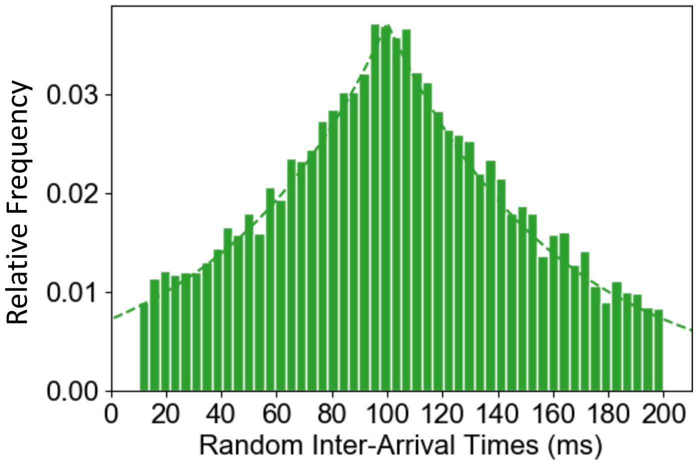
- Pure Laplace distribution is not bounded
- Randomized inter-arrival time must be bounded
- Two ways bound can be enforced:

Truncation



✗ Unbalanced Distribution

Bounding

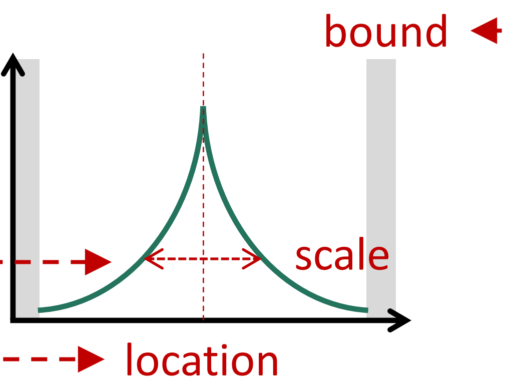


△ Balanced Distribution (with Larger Scale)

# Bounded Inter-Arrival Time Randomized Mechanism

$$\tilde{\mathcal{R}}(\tau_i, j) = \underbrace{\tilde{L}\left(\eta_i(j), \frac{2J_i\Delta\eta_i}{\epsilon_i}, T_i^{\perp}, T_i^{\top}\right)}_{\text{bounded Laplace distribution}}$$

- $Lap\left(\eta_i(j), \frac{2J_i\Delta\eta_i}{\epsilon_i}\right)$ bounded in the range $[T_i^{\perp}, T_i^{\top}]$



# $\epsilon$-Scheduler Model

- Extended Task Model
  - $T_i, \ D_i, \ C_i$

    sets of admissible periods, deadlines and the worst-case execution times

  - $\eta_i, \ T_i^{\perp}, \ T_i^{\top}, \ \Delta\eta_i, \ J_i, \boxed{\epsilon_i}$ — *configurable indistinguishability parameter*

    $\epsilon$-Scheduler extended parameters

- Bounded Inter-Arrival Time Laplace Randomized Mechanism

$$\tilde{\mathcal{R}}(\tau_i, j) = \tilde{L}(\eta_i(j), \frac{2J_i\Delta\eta_i}{\epsilon_i}, T_i^{\perp}, T_i^{\top})$$

# Determining a Feasible $\epsilon$ Value

- Schedule Indistinguishability

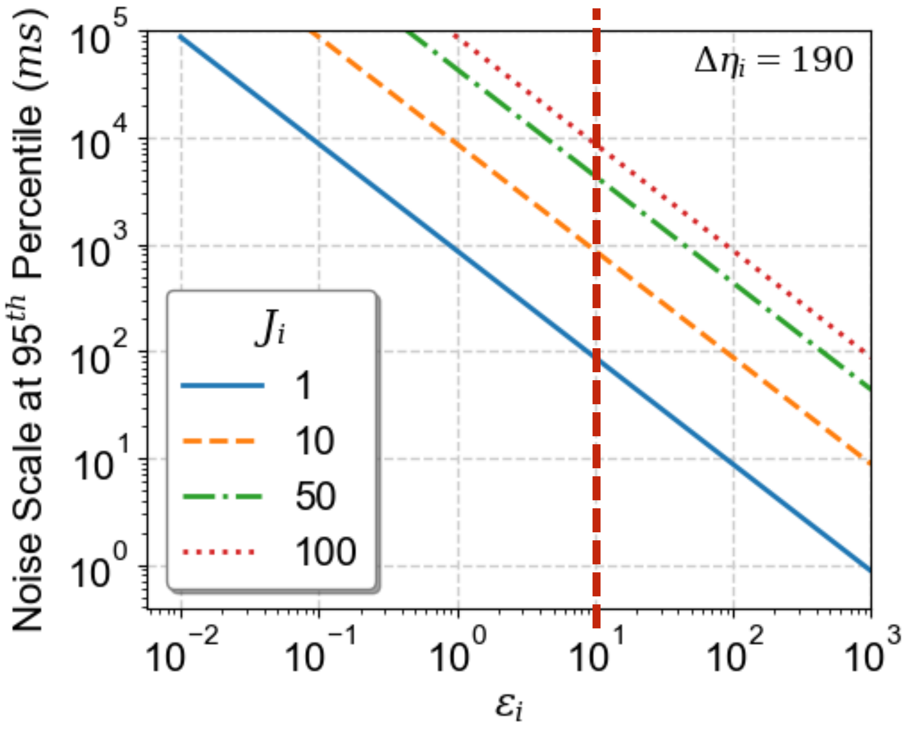
$$\frac{\Pr[\,\mathcal{R}(\tau, j) \in S\,]}{\Pr[\,\mathcal{R}(\tau', j') \in S\,]} \leq e^{\epsilon}$$

$\epsilon$ ↓ Noise ↑ Feasibility in RTS ↓

Example $\epsilon = 0 \quad \Rightarrow \quad \Pr\left[\tilde{\mathcal{R}}(\tau, j) \in S\right] = \Pr\left[\tilde{\mathcal{R}}(\tau', j') \in S\right]$

# Determining a Feasible $\epsilon$ Value (cont.)

- $\epsilon$ vs. Scale of the Noise



# Simulation-Based Performance Evaluation

- Synthetic Task Sets

**6000** Task Sets: Task Set Utilization [0.01,0.1) ... [0.91, 1.0) **10** groups × The Number of Tasks 5, 7, 9, 11, 13, 15 **6** groups × **100**

- Task Parameters

Periods [10ms, 200ms]

$\Delta\eta_i$ $190ms$

$\epsilon_i$ $\{10, 10^3\}$

$J_i = \max(\left\lceil \frac{500ms}{\min(T_i)} \right\rceil \mid \tau_i \in \Gamma)$

- Analysis

DFT-based Analysis

Average Slot Entropy

Inference Precision

QoS Analysis

Scheduling Overhead [☆]

Power Consumption [☆]

[☆] Conducted on RT Linux on real hardware

# Implementation in RT Linux Kernel

- Development Platform

Raspberry Pi 4
Model B

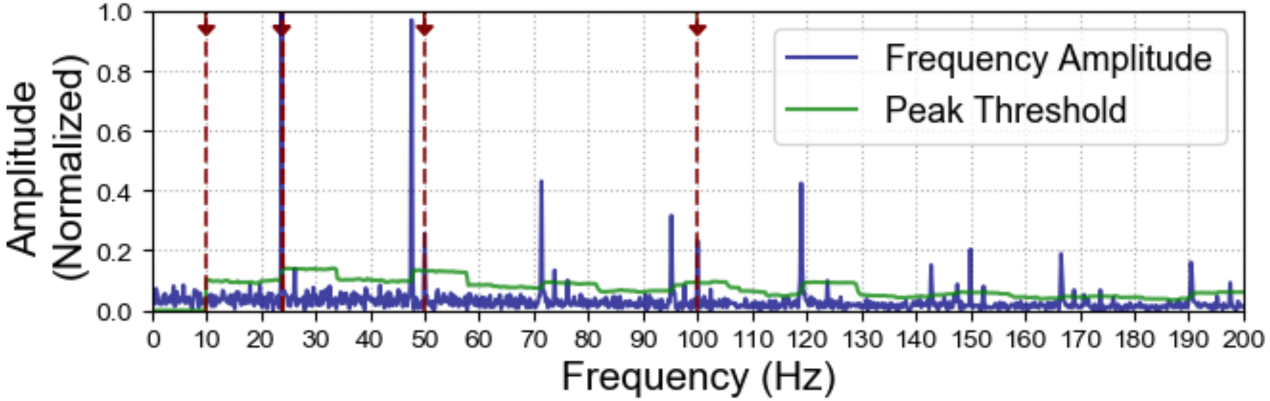
+

Raspbian Linux Kernel
with PREEMPT_RT patch
(4.19.71-rt24-v7l+)

- Scheduler Implementation
  - $\epsilon$-Scheduler was implemented as a scheduling mode in SCHED_DEADLINE

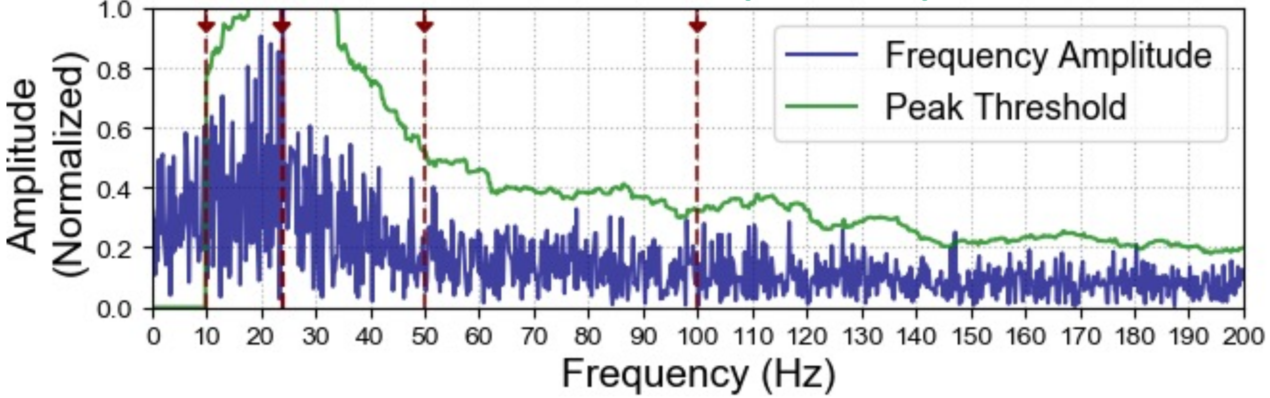
# Discrete Fourier Transform-based Analysis

**Vanilla EDF**

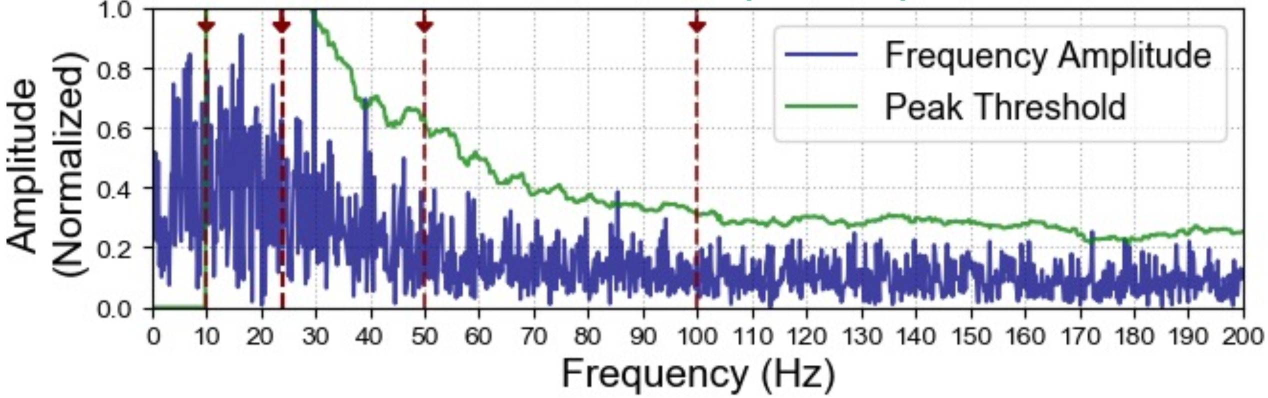


| Task Name | WCET (ms) | Period (ms) | Freq. (Hz) |
|---|---|---|---|
| Software Control Task | 2 | 20 | 50 |
| Mission Planner | 0.002 | 100 | 10 |
| Encryption | 3 | 42 | 23.8 |
| Image Encoding | 18 | 42 | 23.8 |
| Image I/O | 1.46 | 42 | 23.8 |
| Network Manager | 0.03 | 10 | 100 |

**$\epsilon$-Scheduler ($\epsilon = 10^3$)**



**$\epsilon$-Scheduler ($\epsilon = 10$)**



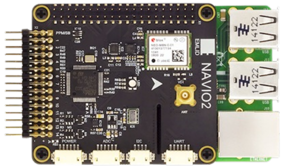
# Summary of Evaluation Results

Compared to Vanilla EDF

| | Periodicity | Average Slot Entropy | Inference Precision | RTS QoS | Scheduling Overhead | Power Consumption |
|---|---|---|---|---|---|---|
| Task Randomization | ↓ | ↑ | ↓ | △ | ↑↑ | ↑ |
| $\epsilon$-Scheduler ($\epsilon = 10$) | ↓↓ | ↑↑ | ↓↓ | ↓↓ | ↑ | △ |
| $\epsilon$-Scheduler ($\epsilon = 10^3$) | ↓↓ | ↑↑ | ↓↓ | ↓ | ↑ | △ |

# Evaluation on Real Applications

- Autonomous Rover System
  - ➢Platform



RPi 4 + Navio2

+

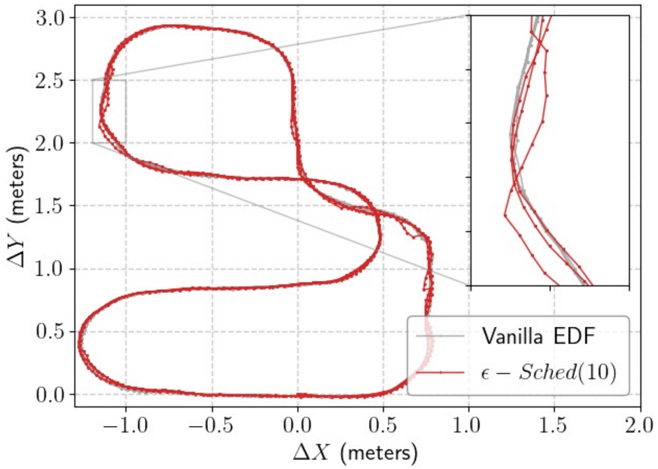
**RoverBot**

autopilot stack

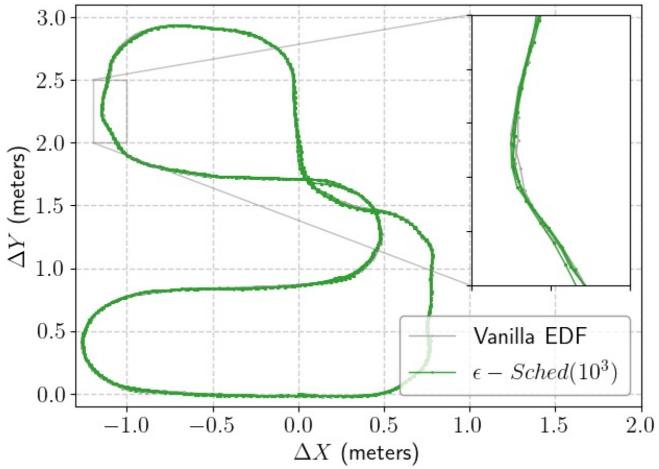
+

  - ➢Impact on Trajectories

$\epsilon$-Scheduler ($\epsilon = 10$)

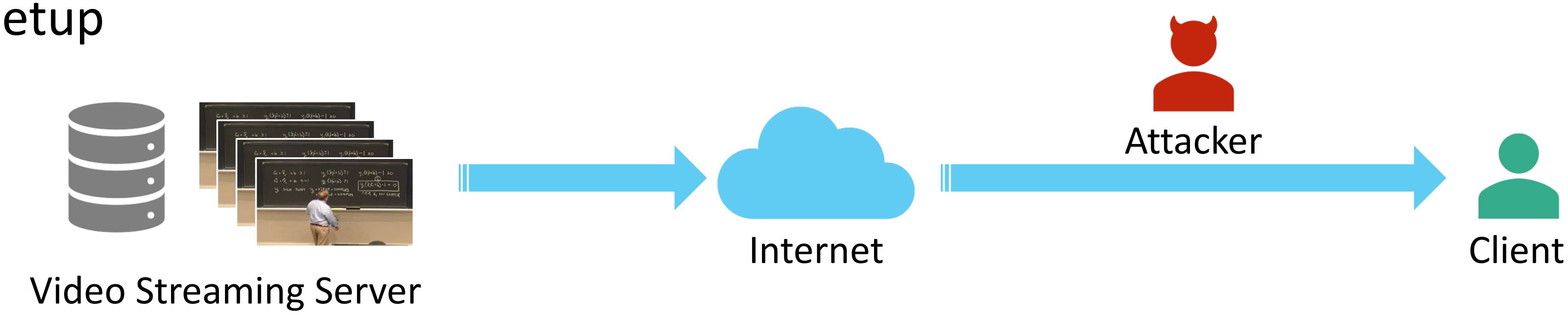


$\epsilon$-Scheduler ($\epsilon = 10^3$)



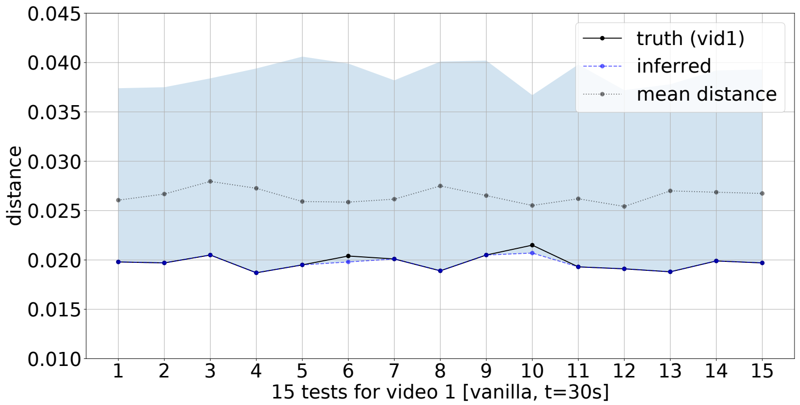
# Evaluation on Real Applications
(cont.)

- Video Streaming over the Internet
  - ➢Setup

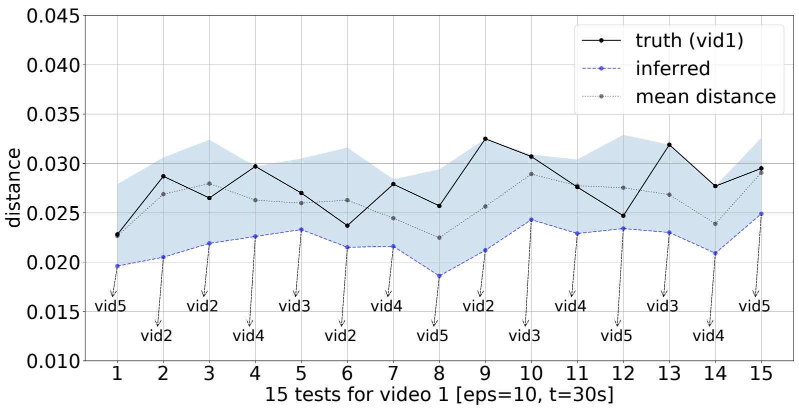


  - ➢Attacker’s Inference Results

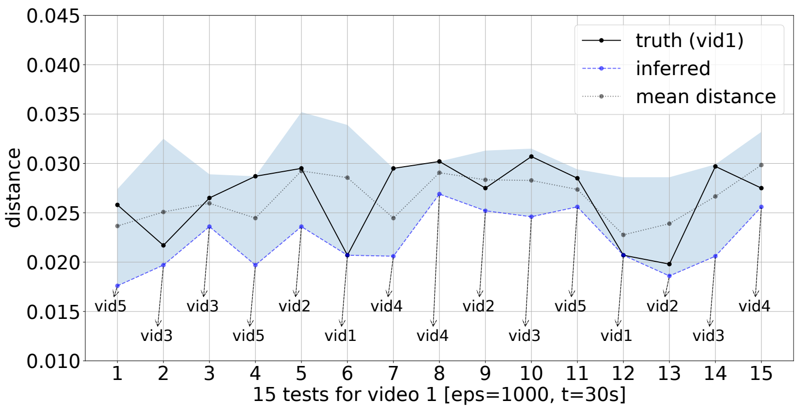
Vanilla EDF



$\epsilon$-Scheduler ($\epsilon = 10$)



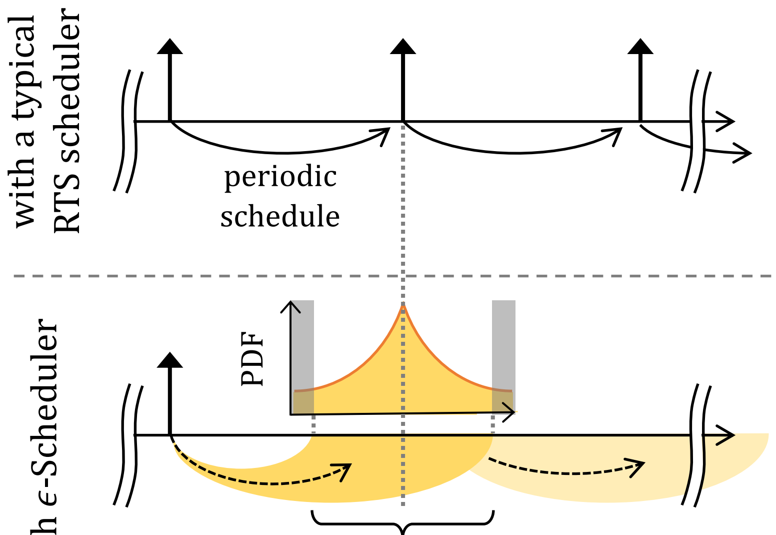
$\epsilon$-Scheduler ($\epsilon = 10^3$)



# Conclusion

## Insight into $\epsilon$-Scheduler

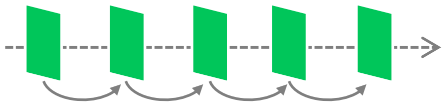
*Illustration of a Task's Schedule*



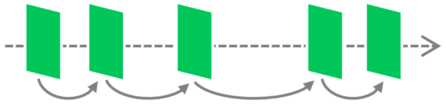
possible arrival time instants
based on a bounded Laplace distribution

## Defense Enabled by $\epsilon$-Scheduler

*Breaking Periodicity*



periodic execution pattern



diversified execution pattern

*Obstructing Predictability*

| | | | | | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 2 | 2 | 3 | 3 | 1 | 2 | 2 | | | 1 | | 2 | 2 | | 1 | 3 | 3 | 2 | 2 | 1 | |
| 1 | 2 | 2 | 3 | 3 | 1 | 2 | 2 | | | 1 | | 2 | 2 | | 1 | 3 | 3 | 2 | 2 | 1 | |
| 1 | 2 | 2 | 3 | 3 | 1 | 2 | 2 | | | 1 | | 2 | 2 | | 1 | 3 | 3 | 2 | 2 | 1 | |
| 1 | 2 | 2 | 3 | 3 | 1 | 2 | 2 | | | 1 | | 2 | 2 | | 1 | 3 | 3 | 2 | 2 | 1 | |
| 1 | 2 | 2 | 3 | 3 | 1 | 2 | 2 | | | 1 | | 2 | 2 | | 1 | 3 | 3 | 2 | 2 | 1 | |

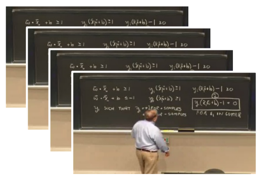
deterministic, predictable schedule

| | | | | | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 2 | 1 | 2 | | 3 | 1 | 3 | 2 | 2 | | 1 | | | 2 | 2 | | 1 | | 2 | | | 1 |
| 1 | | 2 | 2 | | 3 | 2 | 2 | | 2 | 3 | | 1 | 2 | 2 | | 3 | 1 | 3 | 2 | 1 | 2 |
| 3 | 2 | 1 | 3 | 2 | 1 | | | 2 | 2 | | 1 | 2 | 2 | | 1 | | | 2 | 2 | | 3 |
| 3 | 2 | 2 | | 1 | 3 | 1 | | | 2 | 1 | 2 | | | | 2 | 2 | 3 | 1 | 2 | 3 | 2 |
| 2 | | | 1 | 2 | | 1 | | 2 | 3 | 2 | 3 | 2 | 1 | | 2 | 1 | | 3 | 3 | 2 | 2 |

indistinguishable schedule

## Demonstrative Applications

Autonomous
Rover



Video
Streaming