Understanding the Pitfalls of a Differentially Private Fixed-Priority Real-Time Scheduler

Mohammad Fakhruddin Babar, Tamim Ahmed, and Monowar Hasan
School of EECS, Washington State University, Pullman, WA, USA
Email: m.babar@wsu.edu, tamim.ahmed@wsu.edu, monowar.hasan@wsu.edu

***Abstract*—The deterministic nature of real-time scheduling enables strong timing guarantees but exposes systems to schedule-based side-channel attacks. To mitigate such leakage, recent work proposes introducing randomness into scheduling. One such technique is a differentially private scheduler (DPS), which adds noise into task inter-arrival times to achieve "schedule indistinguishability." However, it remains unclear whether such non-deterministic task periods improve overall system security. We find that while DPS is useful for limiting plausible attack instances due to the deterministic periodic nature of task execution, it also *introduces new vulnerabilities that have not been systematically studied before*. We demonstrate that the noise model introduced by DPS increases response-time jitter, which can degrade control stability. In addition, non-deterministic task arrivals alter execution ordering; this, in turn, creates new opportunities for schedule-based attacks that are otherwise infeasible under deterministic scheduling. Our evaluation—based on extensive design-space exploration—shows that DPS can significantly increase the feasibility and success probability of schedule-based attacks compared to traditional fixed-priority scheduling. The findings in this paper highlight a fundamental trade-off between security and real-time requirements and call for the design of schedulers that provide more protection against schedule-based attacks.**

***Index Terms*—ϵ-scheduler, real-time security, scheduling.**

I. Introduction

Cyber-physical systems (CPS) increasingly integrate multiple real-time applications on shared computing platforms, where tasks of different criticality execute concurrently. These systems rely on real-time schedulers to enforce temporal correctness and ensure that safety-critical control tasks meet strict timing constraints. Classical fixed-priority schedulers, such as the rate-monotonic scheduler [1], achieve this goal through deterministic task execution, where job arrivals and scheduling decisions follow predictable patterns.

While determinism is essential for system correctness, it also introduces a fundamental security vulnerability. Adversaries can exploit predictable timing behavior to infer sensitive information or manipulate system execution. Prior work, commonly known as *schedule-based attacks* [2]–[11] has demonstrated that timing observables—such as task arrivals, response times, and execution order—can be leveraged to launch schedule-based attacks. To mitigate such timing channels, researchers propose to introduce *randomness* into scheduling [12]–[19]. Among them, the differentially private scheduler (DPS), also known as the ϵ-scheduler [19], perturbs task inter-arrival times using bounded Laplace noise to achieve *schedule indistinguishability*. DPS aims to limit an attacker's ability to infer timing information by breaking deterministic execution patterns and is the focus of this research. However, till now, *it is unknown how resilient DPS is against schedule-based attacks*. This leads to a central question: ***Does a scheduler designed to prevent timing side-channel attacks inadvertently enable new schedule-based attack opportunities?***

We answer this question through a combination of theoretical analysis and empirical evaluation. We find that removing or reducing determinism comes at a cost. We show that *DPS introduces a new class of vulnerabilities that have not been systematically studied*. Specifically, we demonstrate that the randomness introduced in DPS can increase response-time jitter—leading to control instability—and unintentionally increase the likelihood of some schedule-based attacks. Hence, some attack scenarios may be impossible under deterministic scheduling but become feasible under DPS.

Our Contributions. Our research provides new insights into DPS, a relatively new scheduler designed to protect real-time systems against schedule-based attacks. The goal of this paper is *not* to nullify the contributions of DPS. In fact, our experiments (see §VI-B2) indeed show that the non-deterministic behavior in DPS reduces the number of attack instances for tasksets that are vulnerable under a traditional scheduler (TS). Hence, DPS serves its intended purpose. However, we also find new pitfalls. We both formally and empirically demonstrate that under DPS, it is possible to open new avenues for an attacker to achieve adversarial outcomes that would otherwise be unrealizable for a TS. In particular, we formally show that DPS increases jitter compared to deterministic scheduling and establish conditions under which *(a)* increased jitter reduces control stability and *(b)* lowers the adversarial effort required to trigger instability. We further analyze how variations in execution order in DPS increase the likelihood of some schedule-based attacks compared to TS. We validate our findings using large-scale design-space exploration.

This paper makes the following contributions:

- We formally identify a control-stability vulnerability and show that DPS can destabilize control tasks due to increased jitter (§IV).
- We further demonstrate that DPS increases the likelihood of several schedule-based attacks (§V).
- We conduct thorough design-space exploration and present empirical evaluation showing that DPS widens

the attack surface, even in scenarios where a TS remains non-vulnerable (§VI).

Our implementation is **publicly available** to facilitate reproducibility and support further research [20].

II. Context and Background

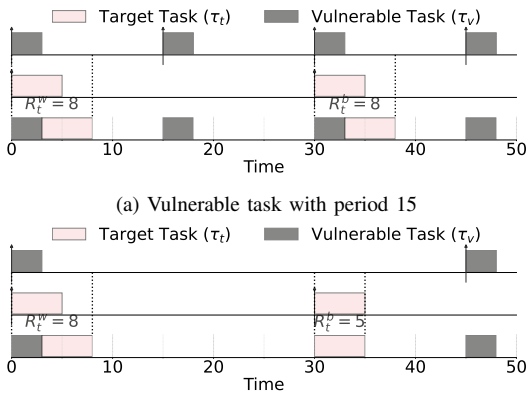
Real-time systems are designed to be *deterministic*—such predictable behavior is necessary to ensure correct system operation and retain temporal guarantees. However, such determinism is a double-edged sword, as it can lead to information leakage [5], [12], [19], [21], [22]. ScheduLeak [6] first identifies a schedule-based attack that enables a malicious task to precisely pinpoint the future execution of a targeted (victim) task. Following the ScheduLeak attack, several other vulnerabilities in real-time schedulers have been identified. Butterfly attack [11] is one of them that can destabilize a low-priority task by manipulating the job release of a high-priority task. Other attacks, identified by Nasri et al. [21], originated from the task execution orders. For instance, an Anterior attack can be triggered if a malicious task is executed just *before* the execution of a victim task. Likewise, a Posterior attack is performed immediately *after* the victim task completes execution. Finally, a Pincer attack combines the anterior and posterior attack situations (i.e., can be triggered when a malicious task executes *before and after* a victim task).

One common strategy to defend against schedule-based attacks is *randomization*. The idea of schedule randomization, first introduced in TaskShuffler [12], is to allow priority inversion by randomly selecting a job from the ready queue, while retaining temporal guarantees for all tasks. However, Nasri et al. [21] discover that randomization is generally *ineffective* against schedule-based attacks and increases the potential for Anterior, Posterior, and Pincer attacks. A more recent study [19] proposes using differential privacy [23] to protect against schedule-based attacks such as ScheduLeak. The core idea is to add *bounded noise* to task arrivals using tools from differential privacy, which makes inter-arrival times less predictable. The authors show that a DPS is more effective than randomization-based defenses (e.g., TaskShuffler) against the ScheduLeak attack.

While prior research [21] investigates the efficacy of randomization-based defenses against schedule-based attacks, *there is no thorough study of how resistant DPS is* to such attacks, particularly those that are identified after ScheduLeak. Unlike prior work that randomizes execution order, DPS perturbs task arrivals, which directly impacts *response-time analysis and control stability*—effects *not* captured in prior study [21]. Therefore, the *primary objective of our paper is to systematically assess the robustness of DPS against more recently discovered schedule-based vulnerabilities*, such as Butterfly, Anterior, Posterior, and Pincer attacks. In the following, we first present a brief overview of those attacks (§II-A) and formally introduce DPS (§II-B).

A. Schedule-Based Attacks

1) Butterfly Attack: Butterfly attack is a novel temporal attack in which adversaries manipulate non-critical tasks to indirectly influence the behavior of critical real-time control applications. This attack aims to disrupt system behavior by affecting timing properties such as *latency* and *jitter*. These timing variations due to the attack can then significantly degrade the *stability* of physical plants. Let L_i denote the latency of a real-time control task τ_i, which is the delay experienced by the actuator (i.e., the best-case response time). The variable J_i denotes the jitter (i.e., the variations in the delay) [24]. Formally, these two metrics are defined as: $L_i = R_i^b$ and $J_i = R_i^w - R_i^b$, where R_i^b and R_i^w are the best-case and worst-case response times, respectively. Throughout this paper we use the term "jitter" to indicate the *response time jitter* [11] which is different than *release jitter* [25] used for some real-time analysis. The control system is *stable* if the following condition holds: $L_i + \alpha_i J_i \leq \beta_i$, where α_i and β_i are constant coefficients dependent on the underlying physical plant.



(a) Vulnerable task with period 15

(b) Vulnerable task with period 45

Fig. 1. Illustration of Butterfly attack. When the malicious task's period is 15 (top figure), the best and worst-case response times of the victim are $R_t^w = R_t^b = 8$. Hence, J_t=0. By changing the inter-arrival time of τ_v from 15 to 45 (bottom figure), the attacker can increase the jitter (e.g., in this illustration, $R_t^w = 8$, $R_t^b = 5$, and hence $J_t = 8 - 5 = 3$).

The Butterfly attack does not require direct manipulation of critical (targeted) tasks or the scheduler. It is sufficient for an attacker to modify the timing behavior of a compromised task, for instance, by triggering successive invocations with inter-arrivals different from its original period, which in turn increases jitter and can destabilize the plant (i.e., violating the stability condition, which is a function of latency and jitter). Let τ_v denote a vulnerable task that can be manipulated by an attacker to trigger the attack, τ_t is the targeted victim task, and T_v and T_t denote the periods of vulnerable and target tasks, respectively. Figure 1 illustrates how the latency and jitter of τ_t are affected when the attacker manipulates τ_v's inter-arrival times.

2) Anterior, Posterior, and Pincer Attacks: Anterior, Posterior, and Pincer attacks are similar in nature in the sense that they are triggered by the execution ordering of a compromised task (one that is controlled by the adversary) and the target (victim) task. For the Anterior case, the attack is performed *before* the execution of a target task. Likewise, a Posterior attack is successful when the compromised task is scheduled *after* the target task completes its execution. The

(a) Anterior attack: τ_v runs immediately before τ_t

(b) Posterior attack: τ_v runs immediately after τ_t

(c) Pincer attack: τ_v is preempted by τ_t

Fig. 2. High-level illustration of Anterior, Posterior, and Pincer attacks.

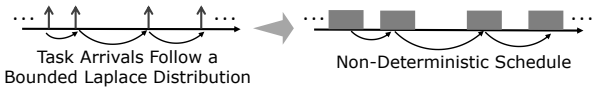


Fig. 3. Task arrivals in DPS follow a bounded Laplace distribution, resulting in a non-deterministic schedule.

Pincer attack requires the compromised task to be executed *before* and *after* the target task. We consider situations where attacker and victim's executions are *adjacent*, as another running task in between them may alter shared state that the compromised task may want to leverage. Figures 2a–2c illustrate these attacks. Examples of attacks include: *(a)* modification of data at a memory address before it is read by a critical task (Anterior), *(b)* data modification after it has been written by another attack (Posterior), and *(c)* access to data by an adversary from a memory address both before and after a critical task modifies data (Pincer).

B. Differentially Private Real-Time Scheduler

The crux of DPS is to add controlled noise in task schedules to break their deterministic execution patterns. In DPS, the task inter-arrivals are non-deterministic as shown in Fig. 3. Let $\eta_i(j)$ denote the inter-arrival time (period) of the j^{th} job of task τ_i. For periodic tasks scheduled by a TS, the function is constant, i.e., $\eta_i(j) = T_i$, for all jobs $j \in \mathbb{N}^+$ of the task. Hence, the inter-arrival times are *deterministic*. DPS adds *noise* Y to the inter-arrival times, i.e., $\eta_i(j) + Y$, where Y follows a (bounded) Laplace distribution centered around the original period. As a result, the schedule becomes "ϵ-indistinguishable" for Q_i job instances.

A Laplace distribution is presented as $\text{Lap}(\mu, b)$, where μ is a location parameter and b is a scale parameter [26]. DPS uses the notion of *inter-arrival time sensitivity*, denoted by $\Delta\eta$, which is the maximum range over which the inter-arrival time could vary when generating inter-arrival times. The location parameter of the Laplace distribution in DPS is set to $\eta_i(j)$. To preserve ϵ-indistinguishability over Q job instances, the scale parameter in DPS is given by: $b = \frac{Q\Delta\eta}{\epsilon}$. Therefore, the randomized inter-arrival time for j^{th} job of τ_i is obtained from a "bounded" Laplace distribution given as follows:

$$\tilde{\mathcal{L}}\left(\eta_i(j), \frac{2Q_i\Delta\eta_i}{\epsilon_i}, T_i^{\perp}, T_i^{\top}\right). \tag{1}$$

In the above equation, the noise is bounded in $[T_i^{\perp}, T_i^{\top}]$, which is based on a pure Laplace distribution $\text{Lap}\left(\eta_i(j), \frac{2Q_i\Delta\eta_i}{\epsilon_i}\right)$. At each new job arrival, the scheduler invokes the bounded Laplace noise function $\tilde{\mathcal{L}}(\cdot)$ presented in Eq. (1) to determine the next job's (non-deterministic) arrival time.

C. Our Research

While the goal of DPS is to prevent schedule-based attacks, such as ScheduLeak, it remains unclear how effective it is against attacks targeting control system stability (e.g., the Butterfly attack). Moreover, because task arrivals are non-deterministic, it remains unexplored to what extent a vulnerable (compromised) task introduces new Anterior, Posterior, and/or Pincer attack potentials as the task execution order changes—attacks that may not otherwise occur under a deterministic scheduler. In contrast to the prior study [21] that explores the usefulness of randomization against schedule-based attacks, our research investigates the robustness of DPS against new classes of attacks, viz., Butterfly, Anterior, Posterior, and Pincer. Because DPS is fundamentally distinct from randomization-based scheduling, it is unclear whether findings from earlier work [21] still hold. Hence, our research revisits schedule-based attacks in the context of DPS and finds that some issues with randomization also hold for DPS.

III. Model and Assumptions

Since we study an existing real-time scheduler (DPS) [19] and its robustness against known attacks (Butterfly, Anterior, Posterior, Pincer) [11], [21], our model and assumptions are identical to those considered in prior research.

A. System Model

We consider a uniprocessor real-time system with n periodic tasks $\Gamma = \{\tau_1, \cdots, \tau_n\}$ scheduled by a differentially private fixed-priority scheduler. Each task τ_i generates an infinite sequence of *jobs*. They have DPS-compliant parameters and are characterized by a tuple $(C_i, D_i, pri_i, T_i, T_i^{\perp}, T_i^{\top})$. Here, C_i denotes the worst-case execution time (WCET), D_i is the deadline, and pri_i is the task priority. The base period is T_i and the parameters $T_i^{\perp}$ and $T_i^{\top}$ denote the range of the inter-arrival times ($T_i^{\perp} \leq T_i \leq T_i^{\top}$), which are obtained from a bounded Laplace distribution as discussed in §II-B. Hence, $T_i^{\perp} \leq T_i[k] \leq T_i^{\top}$, where $T_i[k]$ is the k-th inter-arrival time (i.e., the time between $(k-1)^{\text{th}}$ and k^{th} job of τ_i). Let $A_i[0]$

be the initial release time of τ_i. Subsequent releases are given by: $A_i[k+1] = A_i[k] + T_i[k]$. Note that, for TS, there are no variations in period, i.e., $T_i^{\perp} = T_i^{\top} = T_i$.

The tasks have an implicit deadline ($D_i = T_i$), i.e., they must be completed before their next arrival. We assume task priorities are unique. We use $hp(\tau_i)$ to denote the set of tasks with higher priority than τ_i. A task τ_i can perform control operations and operate a plant P_i, which we refer to as a control task. We consider a linear-quadratic Gaussian (LQG) plant model [27]. We further assume a discrete time clock. In addition, as in all representative papers [11], [19], [21] that form the basis of this work, we conduct a "worst-case" analysis.

B. Response Times and Stability Conditions

The worst-case response time of any job of a task τ_i is obtained by traditional fixed-priority response time analysis [25] as follows: $R_i^w(k+1) = C_i + \sum_{\tau_h \in hp(\tau_i)} \left\lceil \frac{R_i^w(k)}{T_h^{\perp}} \right\rceil C_h$. The recurrence starts with $R_i^w(0) = C_i$ and converges when $R_i^w(k+1) = R_i^w(k)$ for some k. By assumption, the response time $R_i^w = R_i^w(k+1) = R_i^w(k) \leq D_i$. The best-case response time of any job of τ_i is its WCET: $R_i^b = C_i$. For any given job, response time varies between WCET and worst-case time, i.e., $[C_i, R_i^w]$. As noted in §II-A1, the stability of a plant P_i, governed by the task τ_i, is guaranteed if the following condition is satisfied: $L_i + \alpha_i J_i \leq \beta_i$, where L_i is the nominal delay (i.e., the best-case response time) and J_i is the worst-case response-time jitter (difference between worst-case and best-case response times). The condition $R_i^w \leq D_i$ checks the *schedulability* condition, while $L_i + \alpha_i J_i \leq \beta_i$ determines plant *stability*. The combination of these two conditions ensures that the system is both schedulable and stable.

C. Adversary Model

We assume an adversary has similar capabilities as considered in prior work [11], [19], [21]. The attacker manipulates a potentially vulnerable task (τ_v) to cause damage—such as control instability or data manipulation—to a victim task (τ_t). The attacker has no information about the scheduling policy or task parameters. The objective of the adversary is to increase the jitter of the tasks to make the system unstable (Butterfly attack) or trigger an attack at the "right" time by exploiting scheduling artifacts (Anterior, Posterior, and Pincer attacks). We further assume that the vulnerable task τ_v has sufficient processor time, as is the case for other tasks.

IV. Pitfalls of DPS: Control Instability

We start by discussing how non-deterministic task arrivals can negatively affect jitter and control stability (§IV-A). We then present a case study of a DC motor plant (§IV-B).

A. Impact on Jitter and Control Stability

As noted earlier, subsequent jobs of a task in DPS arrive within a bounded interval $T_i^{\perp} \leq T_i[k] \leq T_i^{\top}$ to ensure that the schedule becomes ϵ-indistinguishable. We show that this variation in inter-arrival times can increase jitter for a task (i.e., widen the difference $R^{\mathrm{w}} - R^{\mathrm{b}}$). We now formally quantify the impact of jitter originating from the added noise introduced by DPS. Let us start with the following definition.

Definition 1. *We define* ***ceiling monotonicity*** *property as: for* $\alpha \geq 0$ *and* $0 < a \leq b$, $\left\lceil \frac{\alpha}{a} \right\rceil \geq \left\lceil \frac{\alpha}{b} \right\rceil$.

Suppose each higher-priority task $\tau_h \in hp(\tau_j)$ for a given task τ_j has an inter-arrival time constrained to the range $T_h^{\perp} \leq X_h \leq T_h^{\top}$. Let $\mathbf{T}_j^{\perp} = (T_h^{\perp})_{\forall \tau_h \in hp(\tau_j)}$ and $\mathbf{T}_j^{\top} = (T_h^{\top})_{\forall \tau_h \in hp(\tau_j)}$ denote the vector containing minimum and maximum periods of all tasks with higher priority than τ_j. Besides, let $\mathbf{T}_j = (T_h)_{\forall \tau_h \in hp(\tau_j)}$ denote the vector of periods for TS (i.e., where inter-arrival times for all jobs are fixed). Note that $\mathbf{T}^{\perp} \leq \mathbf{T} \leq \mathbf{T}^{\top}$ (i.e., $T_h^{\perp} \leq T_h \leq T_h^{\top}$). We refer to $J_j^{\mathrm{ts}}(\mathbf{T}_j)$ as *nominal jitter* (i.e., baseline jitter in TS). Further, let us use $J_j^{\mathrm{dps}}(\mathbf{T}_j^{\perp}, \mathbf{T}_j^{\top})$ to denote the jitter in DPS. The following theorem characterizes the jitter for DPS.

Theorem 1. *The jitter experienced by a task* τ_j *in DPS is at least that under TS or greater, i.e.,* $J_j^{\mathrm{dps}}(\mathbf{T}_j^{\perp}, \mathbf{T}_j^{\top}) \geq J_j^{\mathrm{ts}}(\mathbf{T}_j)$.

Proof. For a vector of inter-arrival times $\mathbf{X} = (X_i)_{i \in hp(\tau_j)}$ and a window of size R, let us define the processor requirement for a task τ_j as follows: $\mathcal{F}(R, \mathbf{X}) = C_j + \sum_{i \in hp(\tau_j)} \left\lceil \frac{R}{X_i} \right\rceil C_i$. The response-time bound under $\mathbf{X}$ as the smallest fixed point can be defined as: $R_j(\mathbf{X}) \stackrel{\text{def}}{=} \min\left\{R \geq 0 : \; R = \mathcal{F}(R; \mathbf{X})\right\}$. Therefore, the worst-case and best-case response-time bounds by the smallest fixed points for DPS are defined as: $R_j^w = R_j(\mathbf{T}_j^{\perp})$ and $R_j^b = R_j(\mathbf{T}_j^{\top})$. Let $R_j^w(\mathbf{T}_j)$ and $R_j^b(\mathbf{T}_j)$ denote the best-case and worst-case response times for TS, respectively. The jitter for the differentially private scheduler is given by: $J_j^{\mathrm{dps}}(\mathbf{T}_j^{\perp}, \mathbf{T}_j^{\top}) = R_j(\mathbf{T}_j^{\perp}) - R_j(\mathbf{T}_j^{\top})$. Likewise, the nominal jitter can be calculated as follows: $J_j^{\mathrm{ts}}(\mathbf{T}_j) = R_j^w(\mathbf{T}_j) - R_j^b(\mathbf{T}_j)$.

From the ceiling monotonicity property (see Definition 1), for a given response time R, we can assert the following for $\forall \tau_h \in hp(\tau_j)$: $\left\lceil \frac{R}{T_h^{\perp}} \right\rceil \geq \left\lceil \frac{R}{T_h} \right\rceil \geq \left\lceil \frac{R}{T_h^{\top}} \right\rceil$. Therefore, $\mathcal{F}(R; \mathbf{T}_j^{\perp}) \geq \mathcal{F}(R; \mathbf{T}_j) \geq \mathcal{F}(R; \mathbf{T}_j^{\top})$. Considering this relation, the following conditions hold:

$$R_j^w(\mathbf{T}^{\perp}) \geq R_j^w(\mathbf{T}) \text{ and} \tag{2}$$

$$R_j^b(\mathbf{T}) \geq R_j^b(\mathbf{T}^{\top}). \tag{3}$$

From above, we can conclude $J_j^{\mathrm{dps}}(\mathbf{T}_j^{\perp}, \mathbf{T}_j^{\top}) \geq J_j^{\mathrm{ts}}(\mathbf{T}_j)$. □

With increased jitter in DPS, the risk of control instability increases. The following theorem formally presents the effect of jitter on control stability.

Theorem 2. *The stability margin of DPS is no larger than that of TS.*

Proof. If a task τ_j experienced jitter J_j, its stability margin Δ_j is given by: $\Delta_j = \beta_j - (L_j + \alpha_j J_j)$. The plant becomes unstable when $J_j > \frac{\beta_j - L_j}{\alpha_j}$. Note that when J_j increases, Δ_j decreases (i.e., less stability margin). Let us use J_j^{dps} and J_j^{ts} to denote the jitters experienced by DPS and TS, respectively. As proved in Theorem 1, DPS has a higher jitter (i.e., $J_j^{\mathrm{dps}} \geq J_j^{\mathrm{ts}}$). Thus, when β_j, L_j, and α_j remain unchanged, $\Delta_j^{\mathrm{dps}} = \beta_j -$

$(L_j + \alpha_j J_j^{\text{dps}}) < \Delta_j^{\text{ts}} = \beta_j - (L_j + \alpha_j J_j^{\text{ts}})$, i.e., DPS has a smaller safety margin than that of TS. □

This highlights that *DPS can directly impact physical system safety*. We now show that due to the internals of DPS, the attacker requires minimal (or often no) changes in the vulnerable task's period. As a result, it is possible to achieve Butterfly-like outcomes with less "burden" on the adversary. Consider the target task τ_t is a control task with jitter-margin stability condition $L_t + \alpha_t J_t \leq \beta_t$. The jitter instability threshold is $J_t^{\max} = \frac{\beta_t - L_t}{\alpha_t}$. Let J_t^{ts} and J_t^{dps} denote the jitter of τ_t under TS and DPS, respectively. As Theorem 1 shows, $J_t^{\text{dps}} \geq J_t^{\text{ts}}$ by construction.

Assume an attacker (τ_v) requires a change of period $e \geq 0$ which induces an additional jitter increase $\Delta J_t(e)$ on τ_t by changing τ_v's inter-arrival times, where $\Delta J_t(0) = 0$ and $\Delta J_t(e)$ is nondecreasing in e. We refer to e as the attacker's *work factor*.[1] We define the minimum change required to violate τ_t's stability condition under TS and DPS as:

$$e_{\text{ts}}^* = \min\{e \geq 0 \mid J_t^{\text{ts}} + \Delta J_t(e) \geq J_t^{\max}\} \text{ and} \tag{4}$$

$$e_{\text{dps}}^* = \min\{e \geq 0 \mid J_t^{\text{dps}} + \Delta J_t(e) \geq J_t^{\max}\}, \tag{5}$$

respectively.

We note that a Butterfly attack under TS typically requires an adversary to manipulate the timing behavior of the compromised task to destabilize the target task. In DPS, an adversary often does *not* need to explicitly change the period of τ_v. Due to the scheduler artifacts (i.e., randomized inter-arrival times), there is an increased chance of (implicit) Butterfly attack-like outcomes that an adversary can exploit without deliberate modifications. This results in a less work factor for DPS. The following theorem formally shows that the attacker's work factor in DPS is no more than that in TS.

Theorem 3. *DPS requires a lesser or the same work factor for an adversary to change its inter-arrival time when compared to TS, i.e.,* $e_{\text{dps}}^* \leq e_{\text{ts}}^*$.

Proof. Note that the stability is violated when $J_t > J_t^{\max}$. Hence, for TS, the condition to trigger control instability is $\Delta J_t(e) > J_t^{\max} - J_i^{\text{ts}}$. Likewise, for DPS, $\Delta J_t(e) > J_t^{\max} - J_i^{\text{dps}}$. Since $J_t^{\text{dps}} \geq J_t^{\text{ts}}$, the following condition holds: $J_t^{\max} - J_t^{\text{dps}} \leq J_t^{\max} - J_t^{\text{ts}}$. As $\Delta J_t(e)$ is non-decreasing, achieving the same change in jitter requires a lower work factor for DPS, i.e., $e_{\text{dps}}^* \leq e_{\text{ts}}^*$. □

B. Case Study: DC Motor

We present a case study using a DC motor plant [29] to demonstrate how DPS is susceptible to control instability. We show an *extreme case* where the work factor for an attacker is zero for DPS but non-zero for TS. The plant state is given by the following differential equation: $\dot{x} = \mathbf{A}x + \mathbf{B}u$, where x and u denote the state vector of the DC motor (rotational speed of the rotor and armature current) and the input signal, respectively. The transfer function for this DC motor plant is:

$$P(s) = \frac{\varrho_5}{(\varrho_1 s + \varrho_2)(\varrho_3 s + \varrho_4) + \varrho_5^2}, \tag{6}$$

where ϱ_1 is the motor inertia, ϱ_2 is the friction constant, ϱ_3 is the electric resistance, ϱ_4 is the electric inductance, and ϱ_5 is the torque constant. The values for the motor parameters are: $\varrho_1 = 0.01$, $\varrho_2 = 0.10$, $\varrho_5 = 0.01$, $\varrho_4 = 1.00$, and $\varrho_3 = 0.50$. The matrices A and B are calculated based on the physics of the DC motor and given as follows: $\mathbf{A} = \begin{bmatrix} -10.00 & 1 \\ -0.02 & -2 \end{bmatrix}$ and $\mathbf{B} = \begin{bmatrix} 0 \\ 2 \end{bmatrix}$. Based on the transfer function and control dynamics, the stability curve for this motor is presented in Fig. 4. As the figure shows (dotted vertical line), the stability condition for this plant is $L + \alpha J \leq 20.83$.

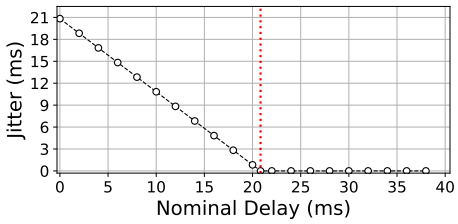


Fig. 4. The stability curve for the DC motor using the transfer function presented in Eq. (6) and LQG controller with a sampling period of 6 ms. The stability threshold is 20.83 (vertical bar).

TABLE I
EXAMPLE TASKSET I

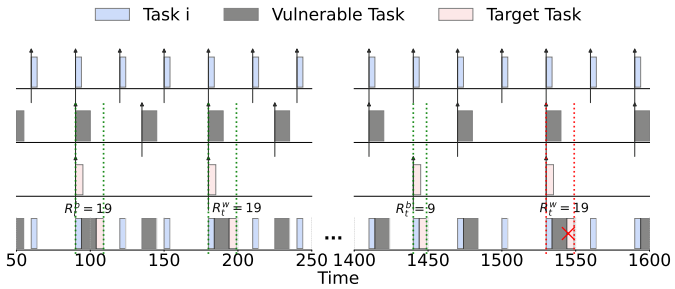
Task	WCET	Period (TS)	Bounds (DPS)
τ_i	4	30	[20, 100]
τ_v	10	45	[20, 100]
τ_t	5	90	[20, 100]

Let us focus on an example that shows how DPS can lead to system instability. Consider a taskset presented in Table I. The taskset has three tasks $\Gamma = \{\tau_i, \tau_v, \tau_t\}$, where τ_i is a general real-time task, τ_v is a potentially vulnerable task, and τ_t is the target task responsible for controlling the DC motor. The priority of the vulnerable task τ_v is higher than the target task τ_t. The execution times for τ_i, τ_v, and τ_t are 4, 10, and 5 units, respectively. For TS, the periods are fixed (30, 45, and 90, respectively). We assume an implicit deadline (i.e., $D_i = T_i$). For DPS, the bounds on the periods for each task τ_j (i.e., $([T_j^{\perp}, T_j^{\top}])$) are $[20, 100]$ units.

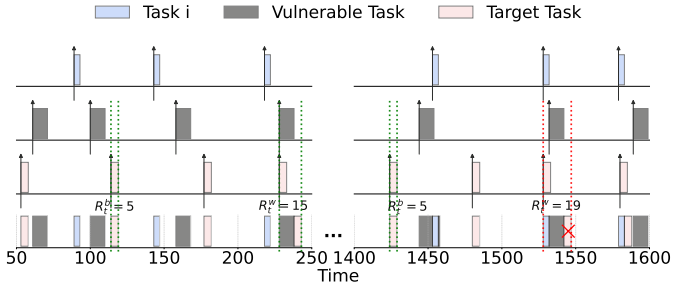
Figure 5a shows the schedule for TS. The best-case response time of τ_t is $L = R_t^b = 19$ (for the job of τ_t that arrives at time 90), and the worst-case response time of τ_t is $R_t^w = 19$ (the job of τ_t arrives at time 180). Therefore, the response-time jitter is $J_t = R_t^w - R_t^b = 19 - 19 = 0$. Substituting the values of L_t and J_t in the stability condition, we get $L_t + \alpha_t J_t = 19 \leq 20.83$. Thus, the system is *stable* (see left side of Fig. 5a).

Now assume the attacker wants to launch a Butterfly attack by changing τ_v's period from $T_v = 45$ to $T_v = 60$ (i.e., $e = 60 - 45 = 15$). The right part of Fig. 5a illustrates this. Hence, the best-case response time of τ_t becomes $L = R_t^b = 9$ (the job of τ_t arrives at 1440) but the worst-case response time of

[1]The term "work factor" is used in security literature to represent the amount of effort needed to successfully trigger an adversarial action [28]. Since we want to measure how much change an attacker needs in τ_v's periods to cause control instability, we use the same term.



(a) Butterfly attack for TS (changing period of τ_v makes the system unstable (corresponding time instance marked by a cross)



(b) Control instability under DPS—the attacker does not require deliberate change of periods

Fig. 5. Stability analysis for a DC motor plant for TS (top) and DPS (bottom). Due to a smaller stability margin in DPS, the system becomes unstable even without requiring the adversary to manipulate the periods of a vulnerable task

τ_t remains unchanged (i.e., $R_t^w = 19$; see the job of τ_t that arrives at time 1530). Therefore, the response-time jitter is $J_t = R_t^w - R_t^b = 19 - 9 = 10$. Note that $\Delta J_t(e) = 10 - 0 = 10$. Substituting these values of L and J, the stability relations: $L_t + \alpha_t J_t = 21 > 20.83$. Thus, the system is *not stable*, and hence the attack is successful.

We next study DPS. Recall from Table I that each task's inter-arrival time is non-deterministic but bounded within $[20, 100]$ units. The best-case response time of τ_t is $L = R_t^b = 5$ (see the job of τ_t arrives at time 114), and the worst-case response time of τ_t is $R_t^w = 15$ (the job of τ_t that arrives at time 228). Therefore, the response-time jitter is $J_t = R_t^w - R_t^b = 15 - 5 = 10$. Substituting these values of L_t and J_t, we get $L_t + \alpha_t J_t = 17 < 20.83$. So, the system is *stable* until this time (left part of Fig. 5b).

Let us consider another window (on the right side of Fig. 5b) in a later interval—we use a large interval to show that the instability condition may be triggered over a longer period. In this case, the best-case response time of τ_t remains unchanged (see the job of τ_t arrives at time 1424). But the worst-case response time of τ_t increased to 19 (see the job of τ_t arrives at time 1528). Thus, the response-time jitter is $J_v = R_t^w - R_t^b = 19 - 5 = 14$. Substituting these values of L_t and J_t, the stability relation becomes $L_t + \alpha_t J_t = 21.80 > 20.83$. So, the system is *not stable*. Since the attacker does not need to change the periods, $e = 0$ and $\Delta J_t(e) = 0$ (i.e., zero burden on the attacker). This is an inherent artifact of DPS that reduces stability margin, as also shown in Theorem 2. As a result, the attacker does not need to change its inter-arrival times, unlike TS (i.e., there is a reduction of the adversary's work factor). This effect is also formally captured in Theorem 3.

V. Implications of Non-Deterministic Periods

We now shift the focus to other classes of attacks that depend on when a critical task executes relative to a vulnerable task. As we noted earlier, the success of Anterior, Posterior, and Pincer attacks depends on the relative execution order of the vulnerable and target tasks. We hypothesize that, as task releases are randomized in DPS, there is a greater chance that a vulnerable task will execute before (Anterior) or after (Posterior) the target task, or be preempted by (Pincer) the target task. We next formally validate our claim.

A. Increased Attack Likelihood

In DPS, the inter-arrival times vary from job to job. This results in the relative offsets of tasks to *drift* over time, which can accumulate across successive jobs. For each job k of τ_v, let $s_v[k]$ and $f_v[k]$ denote its start and finish time. We define the release-phase offset (i.e., the time difference between their k^{th} releases of the τ_v and l^{th} releases of the τ_t) between two adjacent target and vulnerable jobs as $\phi_k = A_t[l] - A_v[k]$. Let $\delta_k = |\phi_k|$. Therefore,

$$\begin{aligned}\delta_{k+1} &= |A_t[l+1] - A_v[k+1]| \\ &= |(A_t[l] + T_t[l]) - (A_v[k] + T_v[k])|. \end{aligned} \tag{7}$$

Substituting $A_t[l] - A_v[k] = \phi_k$, we obtain the following:

$$\delta_{k+1} = |\phi_k + T_t[l] - T_v[k]|. \tag{8}$$

Note that in TS, $T_t[l]$ and $T_v[k]$ are constant for all jobs $l, k = 1, 2, \cdots$ (i.e., a fixed offset). Although the drift per job is bounded by $T_t^{\perp} - T_v^{\top} \le T_t[l] - T_v[k] \le T_t^{\top} - T_v^{\perp}$, as shown in Eq. (8), drifts accumulate over time. As we shall present next, such release offsets create possibilities for Anterior, Posterior, or Pincer attack situations.

Let $\sigma(t)$ denote the task scheduled to be executed at time t. Thus, $\sigma(t) \in \Gamma$ when a task runs, and $\sigma(t) = \tau_{\text{idle}}$ when the processor is not serving any task, where τ_{idle} denotes the idle task. That is, $\sigma(t) \in \Gamma \cup \{\tau_{\text{idle}}\}, \forall t = 0, 1, 2, \cdots$. An attack is possible when a job of τ_v is scheduled closer to τ_t's job. Hence, if τ_v executes in the last non-idle time slot before τ_t begins execution, an Anterior attack can be launched. Likewise, when τ_v executes in the first non-idle time slot after some job of τ_t completes execution, the adversary can trigger a Posterior attack. Besides, when τ_v is preempted by the τ_t, a Pincer attack can be initiated. As noted in §II-A2, we primarily consider adjacency-based attacks. Thus, τ_t must be executed in the last non-idle time slot after and before the τ_v is executed for Anterior and Posterior attacks, respectively, or must preempt τ_v for Pincer attack. This relative order of task execution between τ_t and τ_v provides the attacker with a direct opportunity to manipulate the victim's execution. [2]

Let us formally define Anterior, Posterior, and Pincer attacks in the context of DPS. We first introduce some notations. Let

[2]While non-adjacency-based attacks are possible, they are more coarse as other tasks running between target and victim may alter shared states; and hence, we exclude them in this work.

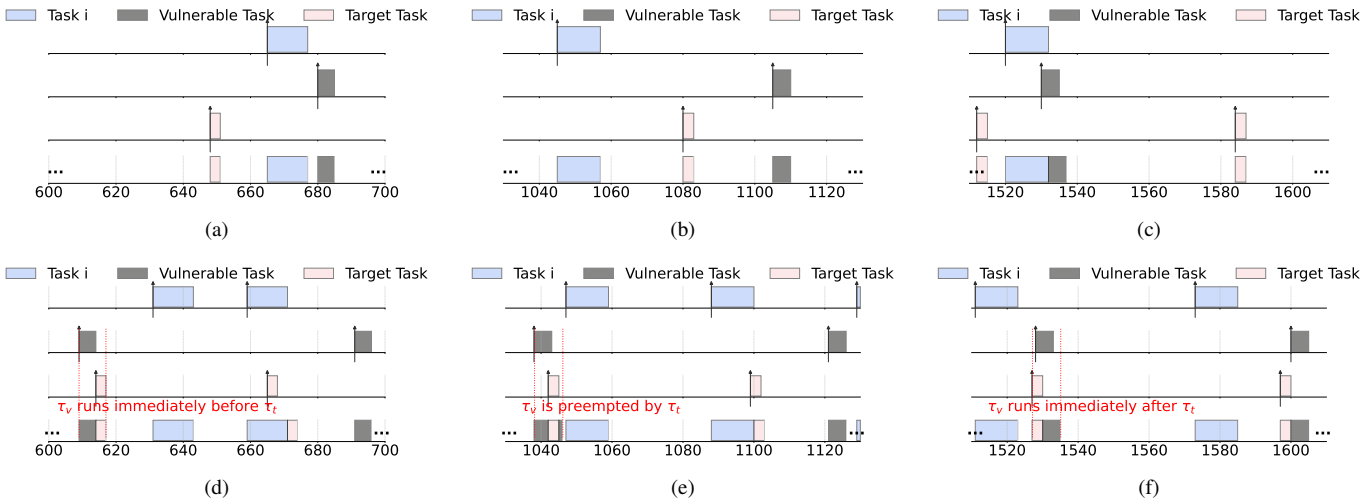


Fig. 6. Illustration of Anterior, Posterior, and Pincer attacks for DPS and TS. Figures (a)–(c) show the schedule for TS, and (d)–(f) are one possible schedule under DPS. As figures (a)–(c) illustrate, no Anterior, Posterior, or Pincer attack possibilities under TS. In contrast, DPS opens possibilities for all of Anterior (d), Posterior (f), and Pincer (e) attacks.

$t_k^+ = \min\{\, t \geq f_v[k] \mid \sigma(t) \neq \tau_{\text{idle}} \,\}$ denote the first non-idle slot at or after the completion of the k-th job of the vulnerable task τ_v. Likewise, let $t_k^- = \max\{\, t < s_v[k] \mid \sigma(t) \neq \tau_{\text{idle}} \,\}$ denote the last non-idle slot strictly before the start of the k-th job of τ_v. Besides, let $[s_v[k],\ f_v[k])$ denote the execution window of the k^{th} job of τ_v. An Anterior attack is possible if $\sigma(t_k^+) = \tau_t$. An attacker can launch a Posterior attack when $\sigma(t_k^-) = \tau_t$. A Pincer attack is possible if there exists a non-idle time slot t_k such that $s_v[k] \leq t_k < f_v[k]$, $\sigma(t_k) \neq \tau_{\text{idle}}$, and $\sigma(t_k) = \tau_t$.

Since DPS varies the inter-arrival times, there are more instances in which these three conditions hold than in TS. We now formally illustrate this phenomenon. Let us start with the following definition.

Definition 2. *For a finite time window H, we define the* ***schedule trace*** *as: $S_{trace}(H) := (\sigma(0), \sigma(1), \ldots, \sigma(H-1))$.*

For a given taskset, let $\mathcal{S}^{\text{ts}}(H)$ and $\mathcal{S}^{\text{dps}}(H)$ denote the set of all traces within window H for TS and DPS, respectively. Further, let $\text{hp} = \text{LCM}(T_1, \ldots, T_n)$ denote the *hyperperiod* of set of tasks, where the $\text{LCM}(x_1, \cdots, x_k)$ operator computes least common multiple of input $x_1, \cdots, x_k$. For each integer $m \geq 0$, we define the m^{th} hyperperiod window as: $W_m = [m \times \text{hp}, (m+1) \times \text{hp}]$, and W_0 denotes the first hyperperiod window. Besides, let $S_{trace}(W_m)$ denote the trace of W_m.

The following theorem characterizes the schedule trace for DPS and TS and asserts that DPS has more task execution-order combinations than TS.

Theorem 4. *If there exists at least one task in DPS with two distinct (feasible) inter-arrival times for a given time window $H > \text{hp}$, then: $\left|\mathcal{S}^{\text{dps}}(H)\right| > \left|\mathcal{S}^{\text{ts}}(H)\right|$.*

Proof. Since each task has a fixed period T_i under TS, release time of each task τ_i is deterministic (i.e., $A_i[k] = A_i[0] + kT_i$). Therefore, at every time instant t, the set of ready jobs is uniquely determined. As a result, the scheduler selects a unique task $\sigma(t)$ at each t. Let us consider the first hyperperiod window W_0. Now, at each epoch of this window $t = 0, 1, \ldots, \text{hp} - 1$, the trace $S_{trace}(H) = (\sigma(0), \ldots, \sigma(\text{hp} - 1))$ is uniquely determined. Again, for another given hyperperiod window W_m, the trace is identical as the task period is deterministic (i.e., the schedule *repeats* after each hyperperiod). Therefore,

$$\sigma(t + m \times \text{hp}) = \sigma(t), 0 \leq t \leq \text{hp} - 1 \text{ and } m = 1, 2, \cdots, \quad (9)$$

i.e., $S_{trace}(m) = S_{trace}(0)$, $\forall m > 0$. Which implies that the schedule trace in TS is *unique* for any window $H > \text{hp}$:

$$\left|\mathcal{S}^{\text{ts}}(H)\right| = 1. \quad (10)$$

Now, let us study the case for DPS. Consider a task τ_j whose k^{th} and l^{th} jobs belong to two different hyperperiod windows and $T_j[k] \neq T_j[l]$. Further assume k^{th} job belongs to W_0 and l^{th} job belongs to W_m. Their arrivals are given by: $A_j[k+1] = A_j[k] + T_j[k]$ and $A_j[l+1] = A_j[l] + T_j[l]$. Since by assumption, $T_j[k] \neq T_j[l]$ in W_m, there exists some time epochs t where $\sigma(t)$ in $S_{trace}(W_0)$ differs from $S_{trace}(W_m)$. This is due to the fact that the schedule in DPS is designed to be non-deterministic by construction. Therefore, we can assert that $\sigma(t + m \times \text{hp}) \neq \sigma(t)$ in DPS. Hence,

$$\left|\mathcal{S}^{\text{dps}}(H)\right| \geq 2. \quad (11)$$

Since $\left|\mathcal{S}^{\text{dps}}(H)\right| > \left|\mathcal{S}^{\text{ts}}(H)\right|$, we conclude that DPS has more variation in the relative task-execution order than TS. □

Based on the above theorem, we now show that DPS is more likely to trigger Anterior, Posterior, and Pincer attack possibilities than a deterministic fixed-priority scheduler.

Let $\eta_v(k)$ denote the inter-arrival time of the k^{th} job of the vulnerable task τ_v. Under TS, the inter-arrival time is fixed, i.e., $\eta_v(k) = T_v, \forall k$. For DPS, the inter-arrival time is perturbed by additive Laplace noise: $\eta_v(k) + Y_v(k) = T_v + Y_v(k)$, where $Y_v(k) \sim \text{Lap}(0, b)$ with scale parameter $b > 0$ (e.g., $Y_v(k)$ is an additive noise to T_v, which makes the location parameter 0, and the parameter b in the noise distribution is $b = \frac{Q_v \Delta \eta_v}{\epsilon}$). Let $\mathcal{A}_k$ denote the attack region for the k^{th} job of τ_v i.e., $\mathcal{A}_k \subseteq \mathbb{R}$. In addition, let $\mathcal{A}_H$ be the

attack region where the attack condition is satisfied for a given observation window H, i.e., $\mathcal{A}_H = \bigcup_{k \in \mathcal{K}(H)} (\mathcal{A}_k)$, where $\mathcal{K}(H)$ denotes the set of job indices of τ_v whose execution intersects with the observation window H.

Let $\widehat{\Gamma}$ contain a set of tasks where Anterior, Posterior, and/or Pincer attacks are possible, by definition (e.g., τ_v runs immediately before/after τ_t and/or preempted by τ_t), regardless of the underlying schedule. The following theorem states that DPS increases attack potential for $\widehat{\Gamma}$.

Theorem 5. *Suppose an attack condition for a set of tasks in $\widehat{\Gamma}$ is never satisfied under TS for a given observation window H, i.e.,* $\Pr^{\text{ts}}(\text{attack in } H) = 0$. *Then, for DPS, there exists a non-zero probability of the attack condition being satisfied for the same window H where an attack is, by definition, possible in $\widehat{\Gamma}$, i.e.,* $\Pr^{\text{dps}}(\text{attack in } H) > 0$.

Proof. By assumption, when tasks in $\widehat{\Gamma}$ are scheduled, there exist situations where attack conditions may trigger (i.e., τ_v runs immediately before/after τ_t and/or preempted by τ_t). Since task arrivals are deterministic in TS, the relative task-execution order is fixed for a given taskset. Assume that the deterministic inter-arrival time T_v of τ_v never falls in any attack region $\mathcal{A}_k$ under TS where the triggering conditions are met. Thus, instantiating an (Anterior, Posterior, and Pincer) attack is impossible under TS within the observation window H, i.e., $\Pr^{\text{ts}}(\text{attack in } H) = 0$.

Under DPS, the inter-arrival time is obtained non-deterministically as $\eta_v(k) + Y_v(k)$, where $Y_v(k)$ follows a Laplace distribution with a probability density function $f_{Y_v(k)}(x) = \frac{1}{2b} \exp\left(-\frac{|x|}{b}\right)$. The density function $f_{Y_v(k)}(x)$ is strictly positive for all $x \in \mathbb{R}$. Now consider an attack feasible interval $I_a(k) = (T_a(k) - \xi(k), T_a(k) + \xi(k)) \subseteq \mathcal{A}_k$, where $T_a(k)$ is the inter-arrival time drawn from a Laplace distribution for the k^{th} release of τ_v where an attack is plausible. The probability that this inter-arrival time of the k^{th} job of τ_v falls inside this interval is given by: $f_{Y_v(k)}(x) \in I_a(k) = \int_{T_a(k)-\xi(k)}^{T_a(k)+\xi(k)} f_{Y_v(k)}(x)\, dx = \int_{T_a(k)-\xi(k)}^{T_a(k)+\xi(k)} \frac{1}{2b} \exp\left(-\frac{|x|}{b}\right) dx$. As $f_{Y_v(k)}(x)$ is strictly positive and the interval $I_a(k)$ has non-zero length, the probability that the inter-arrival time of the k-th job falls within this interval is non-zero, provided that the attack region lies within the bounded distribution. Hence, $\Pr^{\text{dps}}(Y_v(k) \in I_a(k)) > 0$, and since $I_a(k) \subseteq \mathcal{A}_k$, we obtain $\Pr^{\text{dps}}(Y_v(k) \in \mathcal{A}_k) > 0$. This implies that the probability of satisfying the attack condition within the observation window H is non-zero (even if potentially small), i.e., $\Pr^{\text{dps}}(\text{attack in } H) > 0$. Therefore, DPS has the potential to increase the number of Anterior, Posterior, and Pincer attack instances beyond those that are plausible under TS. □

Note that, if an attack is infeasible due to task priority ordering between τ_t and τ_v, neither DPS nor TS has an attack opportunity. For example, for a Pincer attack, τ_v is required to be preempted by τ_t. When τ_v has a higher priority than τ_t, preemption of τ_v is not possible; thus, no Pincer attack—this is also illustrated in our evaluation; see Figs. 7m–7p.

TABLE II
EXAMPLE TASKSET II

Task	WCET	Period (TS)	Bounds (DPS)
τ_i	12	95	[10, 190]
τ_v	5	85	[10, 190]
τ_t	3	72	[10, 190]

TABLE III
KEY PARAMETERS

Parameter	Range
Number of tasks, n	$\{5, 10, 15, 20\}$
Utilization, U	$[0.1, 0.8]$
Execution time, C	$[1, 20]$
Inter-arrival time, T	$[50, 150]$
Period bound $[T^{\perp}, T^{\top}]$	$[10, 190]$
Inter-arrival time sensitivity, $\Delta\eta$	190
Indistinguishability parameter, ϵ	10
Protection duration, Q_i	16
Laplace scale, b	304

B. Illustrative Example

We now show an example where DPS has attack potentials that are not possible in TS. Consider a taskset presented in Table II containing three tasks $\Gamma = \{\tau_i, \tau_v, \tau_t\}$. Their WCETs are $12, 5, 3$, respectively, and periods are $95, 85, 72$, respectively. Task τ_i is the highest-priority real-time task. The vulnerable task that the attacker can control is τ_v, and τ_t is the target task, which has lower priority than τ_v. For DPS, each task's inter-arrival time is bounded by $[10, 190]$.

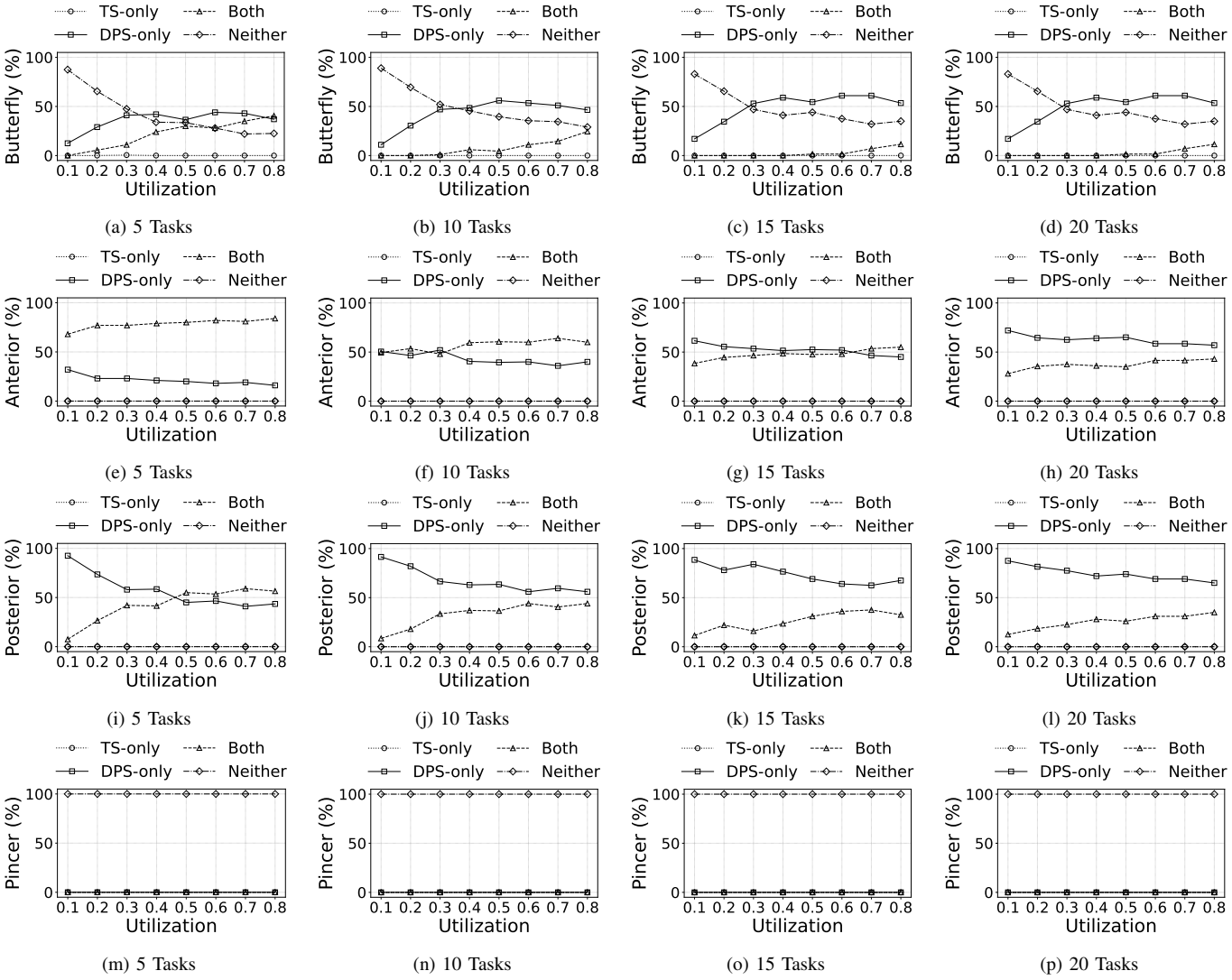
Figures 6a–6c show the corresponding schedule for TS. Likewise, one possible schedule for DPS is presented in Figs. 6d–6f. As Figs. 6a–6c show, the vulnerable task τ_v does not run immediately before the target task τ_t in TS. Thus, there are no Anterior possibilities. In contrast, there exists at least one instance when τ_v runs immediately before τ_t (see time 610 in Fig. 6d), which an adversary can leverage to trigger an Anterior attack. Likewise, as *(a)* τ_v does not run immediately after τ_t and *(b)* τ_v is not preempted by τ_t (as illustrated in Figs. 6a–6c), there are no Posterior and Pincer possibilities when they are scheduled by TS. However, there are instances in DPS when both Posterior (see time 1528 in Fig. 6f) and Pincer (at time 1038, see Fig. 6e) situations happen.

VI. EVALUATION

We conduct a comprehensive empirical evaluation using synthetically generated tasksets to evaluate the robustness of DPS against various schedule-based attacks introduced in §II-A. Our evaluation addresses the following aspects: *(a)* to what extent DPS increases the likelihood of an attack that is (or is not) plausible under DPS; *(b)* how priority of vulnerable and target tasks affects attack likelihood; and *(c)* the relationship between inter-arrival sensitivity ($\Delta\eta$) and attack likelihood. Our implementation is available online [20].

A. Experiment Setup

We used synthetically generated tasksets to assess the robustness of DPS against various schedule-based attacks.



(a) 5 Tasks (b) 10 Tasks (c) 15 Tasks (d) 20 Tasks

(e) 5 Tasks (f) 10 Tasks (g) 15 Tasks (h) 20 Tasks

(i) 5 Tasks (j) 10 Tasks (k) 15 Tasks (l) 20 Tasks

(m) 5 Tasks (n) 10 Tasks (o) 15 Tasks (p) 20 Tasks

Fig. 7. Likelihood of Butterfly, Anterior, Posterior, and Pincer attacks under TS and DPS for $n \in \{5, 10, 15, 20\}$ tasks and varying utilization. In this setup, there are no Pincer attacks as τ_v has higher priority than τ_t. On average, the likelihood of Butterfly, Anterior, Poster, and Pincer attacks for DPS-only instances is 36.62%, 44.90%, and 68.18%, respectively, whereas for Both cases, the values are 17.94%, 55.09%, and 31.81%, respectively.

We varied the system utilization from 10% to 80%. For each utilization, we randomly generated 200 schedulable tasksets. Each taskset contains $n = \{5, 10, 15, 20\}$ tasks. The task periods were randomly selected from $[50, 150]$. Task priorities were assigned according to the rate monotonic order. Following the parameters from prior work [19], the Laplace location parameter was selected from $[50, 150]$ and the scale parameter was set to 304, the lower and upper bounds were set to $T^{\perp} = 10$ and $T^{\top} = 190$, respectively, indistinguishability parameter $\epsilon = 10$, and the protection duration $Q_i = 16$.

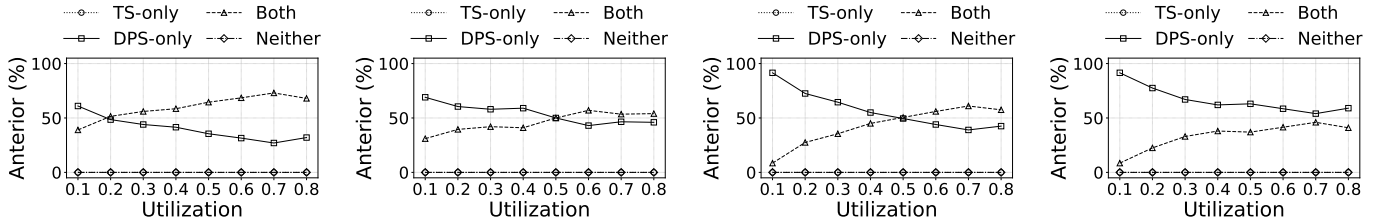
B. Results

1) Experiment #1—Attack Possibilities in Various Tasksets: We define "likelihood of attack" as the ratio of the number of vulnerable taskset (for which an attack is feasible) to the total number of generated tasksets. We study the likelihood of each of the Butterfly, Anterior, Posterior, and Pincer attacks under varying utilizations. For each utilization, we simulated the schedule of each taskset for 200 hyperperiods for both TS and DPS.

For a given taskset Γ, we consider the following four cases.

- TS-only: There is an attack (Butterfly, Anterior, Posterior, and Pincer) likelihood when the tasks in Γ are scheduled under TS; but no attack possibility when they are scheduled using DPS.
- DPS-only: This is the opposite condition when an attack possibility exists under DPS but not in TS.
- Both: Attack possibilities exist for both TS and DPS.
- Neither: The schedules for taskset Γ do not show any instances where the attack is feasible under both of these schedulers.

In the first set of experiments (Fig. 7), we assumed the vulnerable task has a higher priority than the target task, as this is required for the Butterfly attack; subsequent experiments (Fig. 8) analyze various priority orderings for the other three attacks. The key findings of the experiments are that there are 0% TS-only cases, meaning there exists no taskset in which

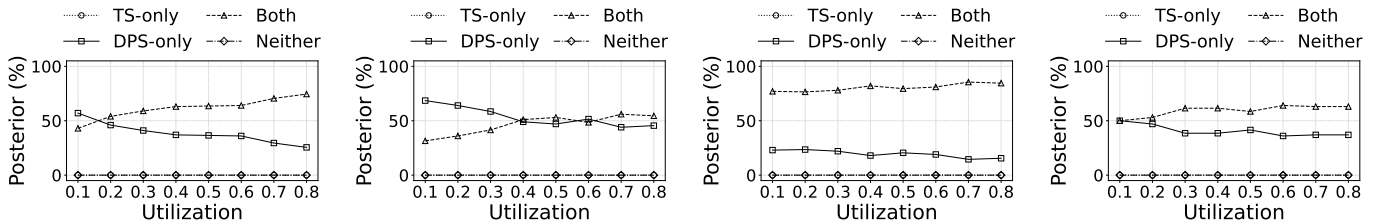


(a) 5 Tasks, priority of τ_v and τ_t randomly assigned

(b) 10 Tasks, priority of τ_v and τ_t randomly assigned

(c) 5 Tasks, τ_v is a lower priority task than τ_t

(d) 10 Tasks, τ_v is a lower priority task than τ_t

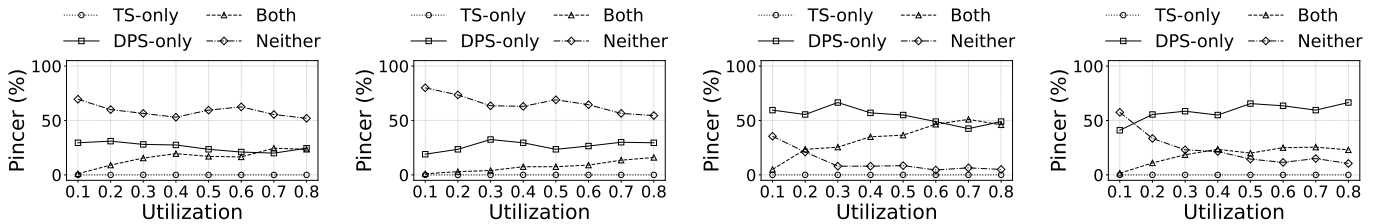


(e) 5 Tasks, priority of τ_v and τ_t randomly assigned

(f) 10 Tasks, priority of τ_v and τ_t randomly assigned

(g) 5 Tasks, τ_v is a lower priority task than τ_t

(h) 10 Tasks, τ_v is a lower priority task than τ_t



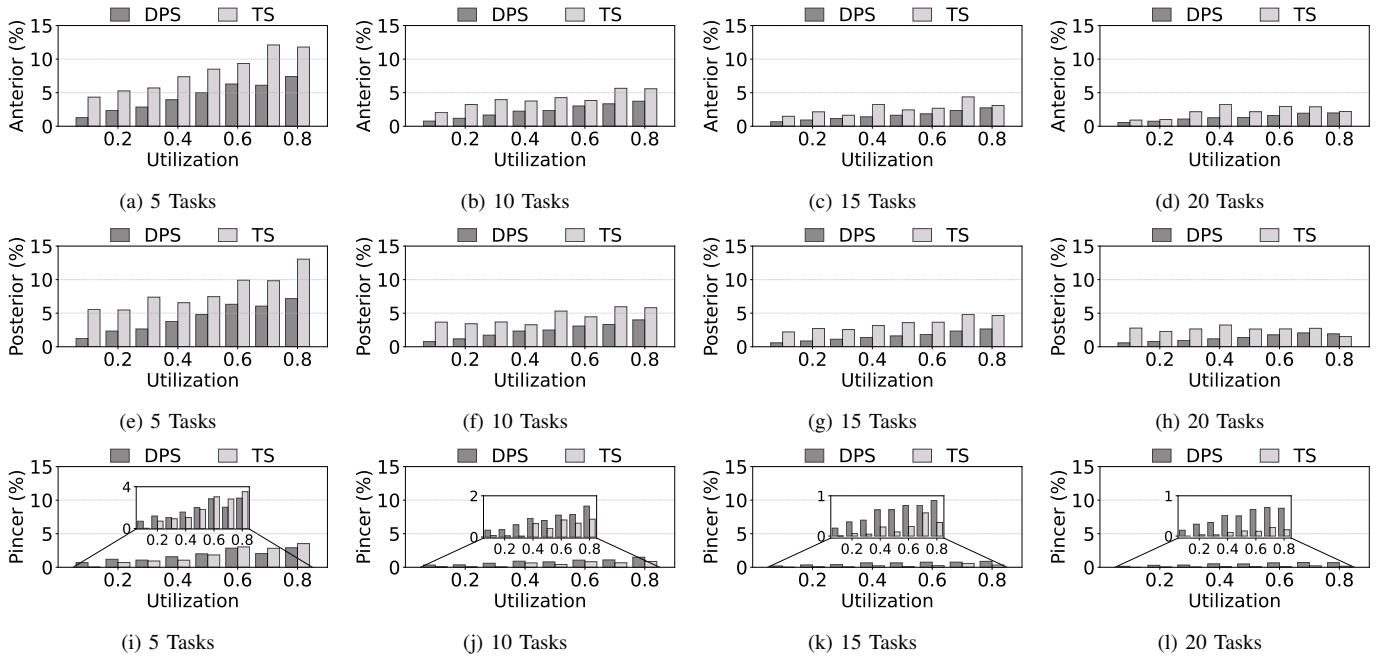
(i) 5 Tasks, priority of τ_v and τ_t randomly assigned

(j) 10 Tasks, priority of τ_v and τ_t randomly assigned

(k) 5 Tasks, τ_v is a lower priority task than τ_t

(l) 10 Tasks, τ_v is a lower priority task than τ_t

Fig. 8. Likelihood of Anterior, Posterior, and Pincer attacks under TS and DPS for $n \in \{5, 10\}$ tasks and varying utilization. When τ_v has a lower priority than τ_t, the likelihood of attacks increases for DPS-only instances.



(a) 5 Tasks

(b) 10 Tasks

(c) 15 Tasks

(d) 20 Tasks

(e) 5 Tasks

(f) 10 Tasks

(g) 15 Tasks

(h) 20 Tasks

(i) 5 Tasks

(j) 10 Tasks

(k) 15 Tasks

(l) 20 Tasks

Fig. 9. Normalized attack likelihood for TS and DPS for $n \in \{5, 10, 15, 20\}$ tasks. Noisy (non-deterministic) arrivals in DPS limit the potential for Anterior and Posterior attacks. However, this is not true for Pincer, as, on average, a vulnerable task may have more opportunities to preempt a target task.

an attack is possible under TS but not under DPS. Hence, if a taskset schedule is vulnerable to any particular type of attack under TS, it is also vulnerable under DPS.

Surprisingly, a high percentage of tasksets fall into the

DPS-only category (where an attack is feasible under DPS but infeasible under TS). This means DPS widens the attack surface, as our analytical findings also show. For instance, consider the numbers when $U = 0.5$ and $n = 5$. For the Butterfly attack, 36.50% of tasksets are in the DPS-only category, compared to 30% in the Both category. Likewise, for Anterior attack, 20% of tasksets are in the DPS-only category, 80% in the Both category, and 0% in the Neither category. For Posterior attack, 45% are in DPS-only, whereas 55% are Both and 0% are in Neither category.

The Butterfly attack likelihood for the DPS-only case is lower in lower utilization. This is because for lower utilization, there is more slack in the system, which prevents the change in period of the vulnerable task from causing control instability for the target task. In addition, attack likelihood decreases for the DPS-only category with higher utilization for Butterfly, Anterior, and Posterior attacks. For higher utilization, the system becomes more constrained to add noise, and behaves more like a deterministic one (TS), and hence, their performance becomes similar. This is why the numbers in the Both category increase.

Another interesting observation is that, for a fixed utilization, the attack likelihood increases with the number of tasks in DPS-only cases, particularly for Anterior and Posterior instances. This suggests that larger tasksets create more opportunities for exploitable temporal relationships under DPS. This is because, with more tasks, the noise model in DPS creates greater uncertainty and more fragmentation in the schedule, leading to more instances where an Anterior or Posterior attack may occur. As Figs. 7m–7p show, there are no instances when a Pincer attack is possible under both TS and DPS. This is because the vulnerable task is assigned a higher priority than the target task. Hence, the target task cannot be preempted by the vulnerable task.

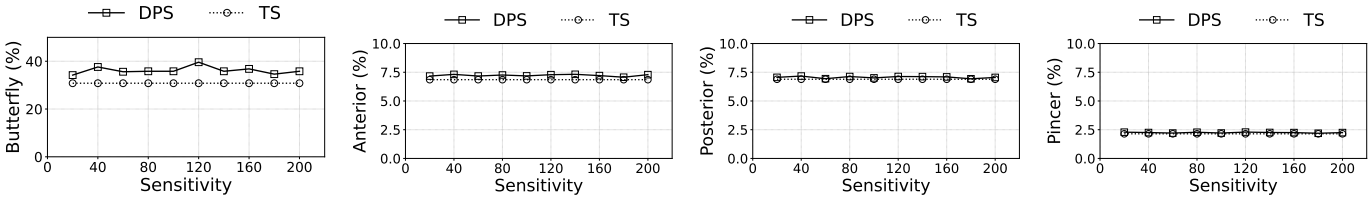
Recall that the Butterfly attack requires vulnerable tasks to have a higher priority than the victim. But there are no such strict requirements for Anterior, Posterior, and Pincer attacks. In the follow-up experiments (Fig. 8), we study attack likelihood with various vulnerable and attacker task priorities. In Fig. 8a–8b, we show likelihood of the Anterior attack with $n = 5$ and $n = 10$ where priority of τ_t and τ_v are selected randomly. Figures 8e–8f and 8i–8j show experiments for Posterior and Pincer attacks, respectively. Similarly, Fig. 8c–8d show the instances for $n = 5$ and $n = 10$, respectively when τ_v has a priority lower than τ_t. Figures 8e–8f and 8i–8j show the likelihood of Posterior and Pincer attacks, respectively, for a similar setup (i.e., τ_v is lower priority than τ_t). When the attacker has lower priority, the likelihood of an attack increases for DPS-only instances. When the attacker has a lower priority than the victim, TS prevents the attacker from executing before the victim and hence no Anterior attacks as identified in prior work [21]. However, DPS perturbs task arrivals and execution ordering, which may allow a lower-priority task to occasionally execute before or around the victim's execution window. As a result, the number of possible overlaps between the attacker and the victim increases, leading to more Anterior, Posterior, and Pincer attack possibilities. The experiments reveal that DPS may unintentionally increase the likelihood of attack success when the attacker has a lower priority than the victim. These behaviors are *consistently observed across large-scale randomly generated tasksets*, indicating that they are not rare corner cases. A similar notion is also noted in our analytical findings.

2) Experiment #2—Likelihood of Attack Instances: Recall that the success of Anterior, Posterior, and Pincer attacks depends on the relative execution order of vulnerable and target tasks. The experiments in Fig. 9 examine instances in which an attack may be initiated under TS and DPS. For this, we count the number of jobs released by the vulnerable task and the target task for 200 hyperperiods for each randomly generated taskset. Note that the schedule repeats after each hyperperiod for TS.

Let N_v^{ts} and N_t^{ts} denote the number of jobs released by τ_v and τ_t, respectively, for a window of time H when they are scheduled under TS. Similarly, let N_v^{dps} and N_t^{dps} denote the corresponding jobs of τ_v and τ_t under DPS. Then, we define the total number of relevant jobs under each scheduler as $N^{\text{ts}} = N_v^{\text{ts}} + N_t^{\text{ts}}$ and $N^{\text{dps}} = N_v^{\text{dps}} + N_t^{\text{dps}}$. Note that N^{ts} and N^{dps} are not necessarily equal, since the job counts may differ between TS and DPS due to noisy arrivals in DPS. Then, for each scheduler (TS and DPS), we count the number of jobs that satisfy the task-execution order for each of the Anterior, Posterior, and Pincer attacks. Let AS^{dps} and AS^{ts} be the number of successful attack instances under DPS and TS. Then we define the normalized attack likelihood as follows: $\chi^{\text{dps}} = \frac{AS^{\text{dps}}}{N^{\text{dps}}}$ and $\chi^{\text{ts}} = \frac{AS^{\text{ts}}}{N^{\text{ts}}}$.

The y-axis of Fig. 9 shows the normalized attack likelihood for various utilizations. The figures show the average values. A key observation is that the average normalized attack likelihood is lower in DPS than in TS, particularly for Anterior and Posterior attacks. The reason being, for a taskset in which both DPS and TS are susceptible to attack (i.e., the Both category for previous experiments), in TS, there are more instances—since they repeated over hyperperiods—where an attack may be plausible. In contrast, DPS breaks this determinism by adding noise, which is the primary reason it was constructed in the first place. In Pincer cases, DPS performs worse, as noisy arrivals can create more fragments in which the vulnerable task is interleaved with the target task (and hence, increases the chances of Pincer attacks).

3) Experiment #3—Impact on Period Bounds: A critical parameter in DPS is inter-arrival time sensitivity, $\Delta\eta$, which determines how the inter-arrival times will vary in the Laplace scale parameter given by $b = \frac{Q\Delta\eta}{\epsilon}$. Our last set of experiments studied the relationship between $\Delta\eta$ and attack likelihood. For this experiment, we used $U = 0.5$ and $n = 10$. We varied sensitivity parameter $\Delta\eta$ in the range $[20, 200]$. Thus, the scale parameter b was varied in the range $[32, 320]$, which is consistent with the prior work [19]. The x-axis of Fig. 10 varies the sensitivity $\Delta\eta$, and the y-axis shows the corresponding likelihood of attack for DPS and TS. Since TS is independent of changes in inter-arrival times, the percentage values are fixed. In our experiment, the attack percentage relatively remained unchanged for DPS. Hence, we conclude that, at least for the chosen parameters borrowed from early work, the attack likelihood does not depend on the sensitivity



(a) Butterfly (b) Anterior (c) Posterior (d) Pincer

Fig. 10. Attack likelihood vs inter-arrival sensitivity. The likelihood of attack does not change significantly with the sensitivity parameter.

TABLE IV
COMPARISON WITH NASRI ET AL. [21].

Dimension	Prior Work	This Research
Scope	Randomized scheduler (TaskShuffler)	Differentially private scheduler (ϵ-scheduler)
Analysis focus	Attack identification and feasibility study	Jitter analysis, attack feasibility study
Control attributes	Not considered	Analysis of control stability degradation
Key contributions	Identification of new attacks (Anterior, Posterior, Pincer) and empirical evaluation	Formal proof and evaluation of robustness against Butterfly, Anterior, Posterior, and Pincer attacks

parameter when DPS draws random inter-arrival times from a bounded Laplace distribution.

VII. RELATED WORK

Real-time systems are by design deterministic, which is necessary for their correct operation. However, such predictability introduces opportunities for several covert-channel [2]–[5] and side-channel [6]–[11] attacks, as identified in prior work.

One way to prevent schedule-based attacks is diversification. Techniques such as TaskShuffler [12], REORDER [13], MAARS [14], and others [15]–[18] explore randomizing task executions to limit determinism, which makes it difficult for an adversary to infer which task executes at a given point in time. As noted in §II, a newer variant of a similar concept is the ϵ-scheduler [19], which is the focus of our research, that introduces bounded noise in task arrival to limit determinism.

A more recent study [30] shows that Butterfly attack arises primarily from a periodic task model and is no longer problematic for sporadic tasks, since the best-case response time is safely bounded. The authors also find that the attack is difficult to implement because it requires precise timing manipulation. After several randomization-based defense techniques were introduced as noted above, researchers found that randomization alone may not be sufficient to protect against schedule-based attacks. Nasri et al. [21] first identified the limitations of randomization. Researchers also explore how to bypass randomization-based defense for hierarchical real-time systems [31].

Unlike prior research that either focuses on attack exploration or on building a new defense mechanism, our work assesses the efficacy of a recent defense mechanism (DPS). Specifically, we explore the robustness of DPS against four recently identified vulnerabilities, viz., Butterfly, Anterior, Posterior, and Pincer attacks. The closest line of work to ours is Nasri et al. [21]. However, there are several differences between the investigation in early work [21] and ours, as summarized in Table IV. Our research is the first to critically evaluate the resilience of DPS against various schedule-based attacks, both analytically and empirically.

VIII. DISCUSSION

Our evaluation across two dimensions (vulnerable-taskset fraction vs. normalized attack-instance rate) will aid designers in making informed decisions. For instance, if an adversary needs the attack condition to be satisfied at least once, DPS significantly broadens the attack potential, since more fractions of tasksets are vulnerable than under TS (as demonstrated in Experiment 1). In contrast, if repeated observations are needed by an adversary, DPS performs well by breaking predictability, which results in a reduced likelihood of attack (see Experiment 2). We should also factor in other practical considerations. For instance, to fully exploit the increased likelihood of attacks in TS, it is critical that the attacker remain stealthy to obtain multiple observations. In addition, while the fraction of the vulnerable taskset metric is useful for broad design-space exploration, designers are often more focused on the specific taskset to be deployed on the target system (hence, the vulnerable-taskset fraction becomes obsolete). Thus, the attack potential of the taskset under consideration should be critically analyzed, as DPS may or may not provide better protection.

Our research also reveals another critical aspect of scheduler-induced control issues (zero-work factor) and attacker-enabled instability (classic Butterfly). For instance, under DPS, an attacker may not need deliberate manipulations to achieve Butterfly-like outcomes. As noted before (§IV), this is an unavoidable scheduler-level artifact of DPS. However, such situations can be avoided by (worst-case) design-time analysis. For instance, for a given taskset, if it is observable that an attacker needs zero work factor for the Butterfly attack, which indirectly implies that, due to scheduling constraints, the plant will become unstable at some point of operation, even if the attacker's compromised task does not perform any adversarial actions. Thus, such tasksets should not be deployed in the first place, or may be further inspected to

determine whether parameters such as task priority, period bounds, etc., should be adjusted to ensure the plant remains stable throughout its operation.

This work studies uniprocessor systems, since both prior attacks and the defense tool (DPS) are designed for single-core platforms. The ideas introduced here can be extended for a partitioned multicore system [32] without a loss of generality if the following conditions hold: *(a)* the tasks are based on the Liu and Layland model [1], *(b)* victim and target tasks are allocated to the same core, and *(c)* there are no shared states between cores. However, the generality of our conclusions to situations where the above conditions do not hold, or to semi-partitioned or global multicore schedulers [33], is *unknown* to us at this point and requires further research, as there is no known DPS model for those schedulers.

We primarily compare the non-deterministic scheduler (TS) with DPS, as DPS claims to be an improvement over TS. Also, we want to study whether the findings from Nasri et al. [21] hold for DPS (which is again a comparison between deterministic and non-deterministic schedulers). Comparing DPS with other randomization methods, including MAARS (Posterior-only) and TaskShuffler, is orthogonal to this work but is a promising direction for future work.

IX. Conclusion

This paper critically examines the resilience of a new real-time scheduler (DPS) designed to protect against various schedule-based attacks. We find that using DPS in a control system can lead to control stability violations. Besides, DPS is also more susceptible to various other schedule-based attacks, such as Anterior, Posterior, and Pincer, and thus faces the same limitations as schedule randomization techniques. We anticipate that our findings will open new research directions to design security-focused real-time schedulers that consider execution-order constraints and control aspects while protecting against scheduler-based attacks.

Acknowledgment

We would like to thank the anonymous reviewers for helping us improve the paper. Besides, we thank Prof. Sibin Mohan and Dr. Chien-Ying Chen for their initial discussion on ϵ-scheduler and for sharing the implementation. This research is partly supported by the United States National Science Foundation CAREER Award 2442595. Any findings, opinions, recommendations, or conclusions expressed in this paper are solely those of the authors and do not necessarily reflect the views of the sponsor.

References

[1] C. L. Liu and J. W. Layland, "Scheduling algorithms for multiprogramming in a hard-real-time environment," *Journal of the ACM (JACM)*, vol. 20, no. 1, pp. 46–61, 1973.

[2] J. Son *et al.*, "Covert timing channel analysis of rate monotonic real-time scheduling algorithm in mls systems," in *2006 IEEE Information Assurance Workshop*, pp. 361–368, IEEE, 2006.

[3] M. Völp, C.-J. Hamann, and H. Härtig, "Avoiding timing channels in fixed-priority schedulers," in *Proceedings of the 2008 ACM symposium on Information, computer and communications security*, pp. 44–55, 2008.

[4] J. Kwak and J. Lee, "Covert timing channel design for uniprocessor real-time systems," in *Parallel and Distributed Computing, Applications and Technologies: 19th International Conference, PDCAT 2018, Jeju Island, South Korea, August 20-22, 2018, Revised Selected Papers 19*, pp. 159–168, Springer, 2019.

[5] M. F. Babar and M. Hasan, "A new covert channel in fixed-priority real-time multiframe tasks," in *2024 IEEE 27th International Symposium on Real-Time Distributed Computing (ISORC)*, pp. 1–6, 2024.

[6] C.-Y. Chen, S. Mohan, R. Pellizzoni, R. B. Bobba, and N. Kiyavash, "A novel side-channel in real-time schedulers," in *2019 IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS)*, pp. 90–102, IEEE, 2019.

[7] M. A. Aguida and M. Hasan, "Work in progress: Exploring schedule-based side-channels in TrustZone-enabled real-time systems," in *2022 IEEE 28th Real-Time and Embedded Technology and Applications Symposium (RTAS)*, pp. 301–304, IEEE, 2022.

[8] S. Liu and W. Yi, "Task parameters analysis in schedule-based timing side-channel attack," *IEEE access*, vol. 8, pp. 157103–157115, 2020.

[9] S. Liu, N. Guan, D. Ji, W. Liu, X. Liu, and W. Yi, "Leaking your engine speed by spectrum analysis of real-time scheduling sequences," *Journal of Systems Architecture*, vol. 97, pp. 455–466, 2019.

[10] S. Hounsinou, M. Stidd, U. Ezeobi, H. Olufowobi, M. Nasri, and G. Bloom, "Vulnerability of controller area network to schedule-based attacks. in 2021 ieee real-time systems symposium (rtss)," 2021.

[11] R. Mahfouzi, A. Aminifar, S. Samii, M. Payer, P. Eles, and Z. Peng, "Butterfly attack: Adversarial manipulation of temporal properties of cyber-physical systems," in *2019 IEEE Real-Time Systems Symposium (RTSS)*, pp. 93–106, IEEE, 2019.

[12] M.-K. Yoon, S. Mohan, C.-Y. Chen, and L. Sha, "Taskshuffler: A schedule randomization protocol for obfuscation against timing inference attacks in real-time systems," in *2016 IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS)*, pp. 1–12, IEEE, 2016.

[13] C. Chen, M. Hasan, A. Ghassami, S. Mohan, and N. Kiyavash, "REORDER: securing dynamic-priority real-time systems using schedule obfuscation," *CoRR*, vol. abs/1806.01393, 2018.

[14] A. Sain, S. Adhikary, I. Koley, and S. Dey, "MAARS: Multi-rate attack-aware randomized scheduling for securing real-time systems," in *Proceedings of the ACM/IEEE 16th International Conference on Cyber-Physical Systems (with CPS-IoT Week 2025)*, pp. 1–12, 2025.

[15] K. Krüger, M. Volp, and G. Fohler, "Vulnerability analysis and mitigation of directed timing inference based attacks on time-triggered systems," *Leibniz International Proceedings in Informatics*, vol. 106, 2018.

[16] A. Samaddar, A. Easwaran, and R. Tan, "Slotswapper: A schedule randomization protocol for real-time wirelesshart networks," *ACM SIGBED Review*, vol. 16, no. 4, pp. 32–37, 2020.

[17] K. Krüger, N. Vreman, R. Pates, M. Maggio, M. Völp, and G. Fohler, "Randomization as mitigation of directed timing inference based attacks on time-triggered real-time systems with task replication," *Leibniz Transactions on Embedded Systems*, vol. 7, no. 1, pp. 01–1, 2021.

[18] H. Baek and C. M. Kang, "Scheduling randomization protocol to improve schedule entropy for multiprocessor real-time systems," *Symmetry*, vol. 12, no. 5, p. 753, 2020.

[19] C.-Y. Chen, D. Sanyal, and S. Mohan, "Indistinguishability prevents scheduler side channels in real-time systems," in *Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security*, pp. 666–684, 2021.

[20] "Understanding the pitfalls of a differentially private fixed-priority real-time scheduler: Implementation." https://github.com/CPS2RL/rt-dps. Accessed: 07-20-2026.

[21] M. Nasri, T. Chantem, G. Bloom, and R. M. Gerdes, "On the pitfalls and vulnerabilities of schedule randomization against schedule-based attacks," in *2019 IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS)*, pp. 103–116, IEEE, 2019.

[22] M. F. Babar, Z. A. H. Hammadeh, M. Hamad, and M. Hasan, "Investigating timing-based information leakage in data flow-driven real-time systems," 2025.

[23] C. Dwork, "Differential privacy," in *Encyclopedia of Cryptography, Security and Privacy*, pp. 649–652, Springer, 2025.

[24] A. Aminifar, P. Eles, Z. Peng, and A. Cervin, "Stability-aware analysis and design of embedded control systems," in *2013 Proceedings of the International Conference on Embedded Software (EMSOFT)*, pp. 1–10, IEEE, 2013.

[25] N. Audsley, A. Burns, M. Richardson, K. Tindell, and A. J. Wellings, "Applying new scheduling theory to static priority pre-emptive

scheduling," *Software engineering journal*, vol. 8, no. 5, pp. 284–292, 1993.

[26] S. Kotz, T. Kozubowski, and K. Podgorski, *The Laplace distribution and generalizations: a revisit with applications to communications, economics, engineering, and finance*. Springer Science & Business Media, 2012.

[27] Y. Tang, Y. Zheng, and N. Li, "Analysis of the optimization landscape of linear quadratic gaussian (lqg) control," in *Learning for Dynamics and Control*, pp. 599–610, PMLR, 2021.

[28] J. H. Saltzer and M. D. Schroeder, "The protection of information in computer systems," *Proceedings of the IEEE*, vol. 63, no. 9, pp. 1278–1308, 1975.

[29] B. Messner and D. Tilbury, "Control tutorials for matlab and simulink-motor speed: System modeling," *CTMS/index. php*, 2019.

[30] S. Baruah, P. Ekberg, M. Hosseinzadeh, A. Li, B. Ward, and N. Zhang, "Who's afraid of butterflies? a close examination of the butterfly attack," in *2023 IEEE Real-Time Systems Symposium (RTSS)*, pp. 53–63, IEEE, 2023.

[31] V. Banerjee, S. Hounsinou, Y. Zhuang, M. Hasan, and G. Bloom, "On evading randomization-based defense in hierarchical real-time systems," *ACM Transactions on Cyber-Physical Systems*, 2025.

[32] J.-J. Chen, "Partitioned multiprocessor fixed-priority scheduling of sporadic real-time tasks," in *2016 28th Euromicro Conference on Real-Time Systems (ECRTS)*, pp. 251–261, IEEE, 2016.

[33] R. I. Davis and A. Burns, "A survey of hard real-time scheduling for multiprocessor systems," *ACM Computing Surveys (CSUR)*, vol. 43, no. 4, pp. 1–44, 2011.