Security-Aware Scheduling of Embedded Control Tasks

VUK LESI, Duke University
ILIJA JOVANOV, Duke University
MIROSLAV PAJIC, Duke University

In this work, we focus on securing cyber-physical systems (CPS) in the presence of network-based attacks, such as *Man-in-the-Middle* (MitM) attacks, where a stealthy attacker is able to compromise communication between system sensors and controllers. Standard methods for this type of attacks rely on the use of cryptographic mechanisms, such as Message Authentication Codes (MACs) to ensure data integrity. However, this approach incurs significant computation overhead, limiting its use in resource constrained systems. Consequently, we consider the problem of scheduling multiple control tasks on a shared processor while providing a suitable level of security guarantees. Specifically, by security guarantees we refer to control performance, i.e., Quality-of-Control (QoC), in the presence of attacks. We start by mapping requirements for QoC under attack into constraints for security-aware control tasks that, besides standard control operations, intermittently perform data authentication. This allows for the analysis of the impact that security-related computation overhead has on both schedulability of control tasks and QoC. Building on this analysis, we introduce a mixed-integer linear programming-based technique to obtain a schedulable task set with predefined QoC requirements. Also, to facilitate optimal resource allocation, we provide a method to analyze interplay between available computational resources and the overall QoC under attack, and show how to obtain a schedulable task set that maximizes the overall QoC guarantees. Finally, we prove usability of our approach on a case study with multiple automotive control components.

CCS Concepts: • **Software and its engineering → Real-time schedulability**; • **Computer systems organization → Embedded and cyber-physical systems**; Embedded systems; Embedded software; • **Security and privacy → Distributed systems security**; *Intrusion detection systems*; • **Theory of computation →** *Linear programming*;

Additional Key Words and Phrases: CPS security, real-time scheduling, quality-of-control, mixed integer linear programming

ACM Reference format:
Vuk Lesi, Ilija Jovanov, and Miroslav Pajic. 2017. Security-Aware Scheduling of Embedded Control Tasks. *ACM Trans. Embedd. Comput. Syst.* 9, 4, Article 131 (October 2017), 21 pages.
https://doi.org/nn.nnn/nnn_n

This work was supported in part by the NSF CNS-1652544 and CNS-1505701 grants, and the Intel-NSF Partnership for Cyber-Physical Systems Security and Privacy. This material is also based on research sponsored by the ONR under agreements number N00014-17-1-2012 and N00014-17-1-2504.
Authors' addresses: V. Lesi, I. Jovanov, and M. Pajic, Department of Electrical and Computer Engineering, Duke University, 100 Science Drive, Durham, NC 27708, USA; emails: {vuk.lesi, ilija.jovanov, miroslav.pajic}@duke.edu.
This article was presented in the International Conference on Embedded Software (EMSOFT) 2017 and appears as part of the ESWEEK-TECS special issue.

1539-9087/2017/10-ART131 $15.00
https://doi.org/nn.nnn/nnn_n

1 INTRODUCTION

Security of embedded control systems and, recently, cyber-physical systems (CPS) has usually been an afterthought. Yet, with the increase in network connectivity and system design complexity these systems have become more susceptible to various types of attacks. This was illustrated in several high-profile incidents including the Stuxnet [12] and automotive (e.g., [11, 13]) attacks. In many cases, such as in automotive industry, these systems rely on perimeter security where internal networks and ECUs are resource constrained, mostly depending on security of the gateway and external communication channels. Yet, when attackers circumvent classical perimeter security barriers, they can have a significant effect on systems' operation [11, 13]. Also, some of the internal system components may be tampered with, allowing the attacker direct access to the internal network [11]. These network-based attacks present a major threat because they enable a *remote attacker* to modify safety-critical messages (e.g., sensor measurements and actuator commands as in [11, 12]) communicated over the low-level network.

In this work, we focus on securing CPS in the presence of network-based attacks, such as *Man-in-the-Middle* (MitM) attacks, where the attacker is able to compromise all or some of the links between the sensors/actuators and controllers; thus, the information delivered to and from the controller may differ from the actual sensor measurements and actuator commands. In addition, as most of these systems are safety-critical, with predefined procedures in case when a fault or intrusion is detected, we consider *stealthy* attacks where the attacker wants to remain undetected until his objective is achieved. As recently shown for a large number of systems, by changing messages from a subset of sensors, a stealthy attacker can force the plant into any (potentially) unsafe state through the actions of the controller [18, 24]. Even for systems for which the set of states where the attacker could force the system is bounded, the attacker could easily move the plant far from the desired reference point; that way, he would significantly degrade control performance and even endanger system safety while remaining stealthy [27, 28].

These results introduce very conservative constraints on the number of sensors that if compromised could cause unsafe system operation or at least significant control degradation. They are also obtained on the assumption that once the communication channel between a sensor and the controller is compromised, the attacker can inject attack signals in all measurements obtained from the sensor. On one hand, some of these network-based attacks can be avoided by ensuring data integrity and authentication with the use of standard cryptographic tools, such as adding message authentication codes (MACs) to communicated measurements. On the other hand, authenticating *all* measurements from a suitable number of sensors (e.g., based on the design frameworks from [21, 33]), incurs a significant computational overhead, making it unsuitable for these usually resource-constrained systems. For instance, computing only a scalar PID controller update takes an order of magnitude less time than computation of a 32-bit MAC — e.g., 12 μs for PID update on 96MHz 32-bit Cortex-M3, and ~100 μs for the MAC computation. Thus, this common approach to 'adding security' into existing and new CPS could prevent schedulability of a number of control tasks that always have to inspect integrity of the incoming data.

Consequently, in this work we consider the problem of scheduling multiple control tasks on a shared processor, while providing a suitable level of security guarantees in the presence of attacks on sensor data delivered to the controllers. Specifically, by security guarantees we refer to control performance, i.e., Quality-of-Control (QoC), in the presence of attacks. While our results can be extended for scenarios where actuator commands can also be compromised, in this paper we focus on defense against false-data injection attacks into sensor measurements only; this is caused by the fact that attacks on commands sent to actuators cannot in general remain stealthy while significantly degrading system performance [18].

We exploit recent results showing that attacker's impact can be significantly limited even when sensor data integrity is only intermittently enforced, for instance by occasionally adding MACs to transmitted sensor measurements [15, 16]. Thus, we start by mapping requirements for QoC in the presence of attacks for each control loop, into constraints for security-aware control tasks; these tasks, in addition to the standard control-related operation, intermittently perform data authentication as part of the controller's execution. This, in turn, enables us to analyze the impact of security-related computation overhead on both schedulability of control tasks and QoC in the presence of attacks.

To achieve this, we transform the problem of scheduling of security-aware control tasks into scheduling of specific multiframe tasks (relaxation of the model presented in [4, 25]). We then introduce a technique to perform schedulability analysis for the task model and show how synthesis of such feasible control task set can be formulated as a mixed-integer linear programming (MILP) problem. In addition, to facilitate optimal allocation of system resources we provide a method to analyze interplay between available computational resources and the overall QoC under attack (i.e., for all control loops). For underutilized systems where the CPU has additional available computation time, we show how QoC under attack can be improved by increasing the integrity enforcement rate for control tasks that maximize the overall QoC. Similarly, if adding new tasks would result in an overutilized (i.e., unschedulable) system, the presented method can be used to optimally reduce the overall QoC under attack, while ensuring task schedulability. Finally, we illustrate the applicability of the proposed techniques both on generic test-cases as well as a realistic automotive case-study.

This paper is organized as follows. In Section 2, we define the problem considered in this work. In Section 3, we present a framework to evaluate QoC in systems with intermittent integrity enforcements. Furthermore, in Section 4 we present our approach to modeling security-aware tasks in these systems, followed by methods for schedule synthesis with predefined QoC requirements (Section 5). Section 6 introduces a technique for synthesis of feasible schedules that maximize QoC guarantees in the presence of attacks. Finally, in Section 7, we evaluate our approach on generic workloads as well as a realistic case study, before providing an overview of related work (Section 8) and concluding remarks (Section 9).

2 MOTIVATION AND PROBLEM STATEMENT

Consider the problem of controlling N discrete-time control systems Σ_i, $i = 1, \ldots, N$, of the form

$$\begin{aligned} \mathbf{x}_i[k+1] &= \mathbf{A}_i\mathbf{x}_i[k] + \mathbf{B}_i\mathbf{u}_i[k] + \mathbf{w}_i[k] \\ \mathbf{y}_i[k] &= \mathbf{C}_i\mathbf{x}_i[k] + \mathbf{v}_i[k] \end{aligned}$$

where $\mathbf{x}_i[k] \in \mathbb{R}^n$, $\mathbf{u}_i[k] \in \mathbb{R}^m$, and $\mathbf{y}_i[k] \in \mathbb{R}^p$ denote state, input and output vectors of the i^{th} plant at time k, respectively. Also, $\mathbf{w}_i \in \mathbb{R}^n$ and $\mathbf{v}_i \in \mathbb{R}^p$ denote the process and measurement noise, distributed as independent identically distributed Gaussian random variables, with covariances $\mathbf{Q}_i$ and $\mathbf{R}_i$, respectively. Since each of the discrete-time plant models is obtained by discretizing the corresponding continuous plant with the sampling time T_{s_i}, we denote the plants as $\Sigma_i(\mathbf{A}_i, \mathbf{B}_i, \mathbf{C}_i, \mathbf{Q}_i, \mathbf{R}_i, T_{s_i})$.

Each system Σ_i is to be controlled by a feedback controller in the general form (e.g., capturing observer-based state feedback)

$$\begin{aligned} \hat{\mathbf{x}}_i[k+1] &= \mathbf{f}_i\left(\hat{\mathbf{x}}_i[k], \mathbf{y}_i^{net}[k]\right) \\ \mathbf{u}_i[k] &= \mathbf{g}_i\left(\hat{\mathbf{x}}_i[k], \mathbf{y}_i^{net}[k]\right), \end{aligned}$$

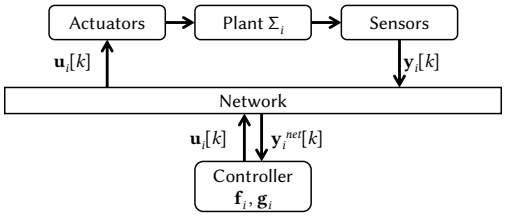


Fig. 1. System architecture for each control-loop.

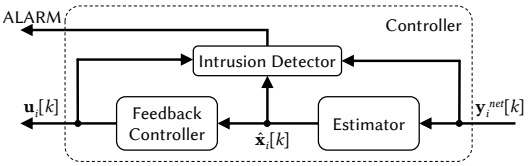


Fig. 2. General controller design.

where $\mathbf{f}_i(\cdot)$ and $\mathbf{g}_i(\cdot)$ denote any linear mappings, $\hat{\mathbf{x}}_i[k]$ is the state of the controller (e.g., estimated plant state) and $\mathbf{y}_i^{net}[k]$ denotes sensor measurements received by the controller in step k over a communication channel/network, as shown in Fig. 1. In general, the controller can be designed using various techniques to ensure, for example, system stability or optimal performance. Finally, each controller is executed as a periodic task $T_i^{ctrl}(c_i^{ctrl}, p_i, d_i)$ on a shared processor, with periods $p_i = T_{s_i}$ and worst-case execution time (WCET) c_i^{ctrl}; to simplify our notation we assume that each task's deadline $d_i = p_i$ and denote the tasks as $T_i^{ctrl}(c_i^{ctrl}, p_i)$.

Without attacks on communication between sensors and the controller, it follows that $\mathbf{y}_i^{net}[k] = \mathbf{y}_i[k]$. However, with MitM attacks, the controller receives values that could potentially differ from the actual sensor measurements, which would cause control performance degradation and potentially unsafe control. To differentiate between system evolution with and without attacks, we add superscript a to the variables affected by the attacker's influence. For instance, the plant's state and outputs in the presence of attacks are denoted as $\mathbf{x}_i^a[k]$ and $\mathbf{y}_i^a[k]$, respectively. Now, attacks on sensor measurements delivered to the controller can be modeled as

$$\mathbf{y}_i^{net,a}[k] = \mathbf{y}_i^a[k] + \mathbf{a}_i[k] = \mathbf{C}\mathbf{x}_i^a[k] + \mathbf{v}_i^a[k] + \mathbf{a}_i[k],$$

where $\mathbf{a}_i[k]$ represents a sparse vector of values injected by the attacker. Sparsity of the vector depends on the set of compromised sensor flows — if communication from a sensor to the controller is not corrupted then the corresponding value in $\mathbf{a}_i[k]$ has to be zero, for all k. This allows us to capture any assumptions about the set of compromised sensor flows (e.g., the number of the flows). However, to simplify our presentation, unless otherwise stated in this paper we present the worst-case scenario, where the attacker can compromise all sensor flows — i.e., measurements from *all* sensors.

Furthermore, in this work, we assume that the attacker can inject any false measurements to be received by the controller (i.e., $\mathbf{a}_i[k]$ can have any value), except at times when data integrity is enforced with the use of standard cryptographic mechanisms (e.g., MACs). Our assumption is that the attacker does not posses shared secret keys used to generate the MACs and thus cannot corrupt those packets,[1] meaning that at these times $\mathbf{a}_i[k] = \mathbf{0}$. We also assume that the attacker has full knowledge of the system, system dynamics and architecture, which would allow him to manipulate the controller to force the system into a desired state, by carefully changing delivered sensor measurements. In addition, the attacker knows the times when the MACs are to be used, and can use the knowledge to plan ahead. Finally, the attacker's goals are: (1) to maximally reduce control performance (i.e., QoC) by manipulating the system into a state that differs from the desired reference point/trajectory, and (2) to remain stealthy — i.e., undetected by the system; hence, in addition to not inserting false data packets in time-frames when MACs are checked, the

[1]In this case, the attacker could potentially prevent these messages from being delivered. However, since Denial-of-Service attacks are easier to detect in these systems that utilize low-level networks with reliable communication protocols, in this work we do not consider such attacks.

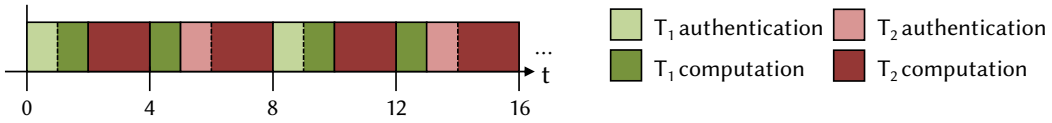


Fig. 3. Scheduling two security-aware control tasks – otherwise infeasible task set $\{T_1^{ctrl'}(2,4), T_2^{ctrl'}(3,4)\}$ becomes schedulable if authentication computations are required in every other period.

falsified sensor measurements should not trigger alarm in the controller's Intrusion Detection System (IDS).

Note that the common practice of enforcing integrity of all communication packets could become infeasible due to additional computation costs. For instance, consider two control tasks $T_1^{ctrl}(1,4)$ and $T_2^{ctrl}(2,4)$ which can be scheduled on a shared CPU. If for each task, authentication of received sensor measurements required to update the controller, results in an increase of the execution time by 1 time unit in every period, the equivalent tasks set with $T_1^{ctrl'}(2,4)$ and $T_2^{ctrl'}(3,4)$ becomes infeasible. On the other hand, as recently shown in [15, 16], even intermittent data integrity enforcement can significantly limit the attackers impact on the system. Therefore, from the perspective of QoC under attack, it may be enough for each of the considered systems to guarantee data integrity for every other control task execution, which would result in a schedulable task set, with a schedule illustrated in Fig. 3.

Consequently, in this paper we focus on tradeoffs between the QoC in the presence of attacks and integrity enforcement overhead for security-aware control tasks, in systems with hard real-time tasks. Specifically, we address the following problems:

- In order to facilitate security-aware scheduling that considers computation overhead due to integrity enforcements, how can we map requirements for QoC in the presence of attacks into constraints for security-aware control tasks?
- How to schedule security-aware control tasks, while ensuring the desired control performance for each of the control loops even in the presence of attacks?
- Is it possible to allocate available resources (i.e., computation time) to each security-aware control task such that the overall (i.e., for all tasks) security guarantees, in terms of control performance under attacks, are maximized?

We start with the recently introduced framework for security-aware control with intermittent data-integrity enforcements.

3 RELAXING INTEGRITY REQUIREMENTS FOR SECURE CONTROL

Common controller architecture, illustrated in Fig. 2, contains a state estimator, feedback controller and, if security is a concern, an IDS. The IDS exploits physical properties of the system (i.e., model of the plant Σ_i and controller $(\mathbf{f}_i, \mathbf{g}_i)$) to raise alarm in the case of attack. Depending on the considered control architecture and noise model, some controllers employ security-aware estimators (e.g., [28]) and set-based IDSs, or standard Kalman filter-based estimators with statistical IDSs (e.g., the χ^2 detector as in [16, 18, 24] or the Sequential Probability Ratio Test (SPRT) detectors as in [15]).

On one hand, by compromising a sufficient number of sensor flows,[2] an intelligent stealthy attacker can use shortcomings of such detectors and system dynamics to force the system away from the desired reference point and significantly reduce control performance. This is achieved by introducing a state estimation error that deceives the controller into applying unsuitable actuation

[2]The exact number depends on the utilized control architecture and noise model. More details can be found in [18, 28] and references therein.

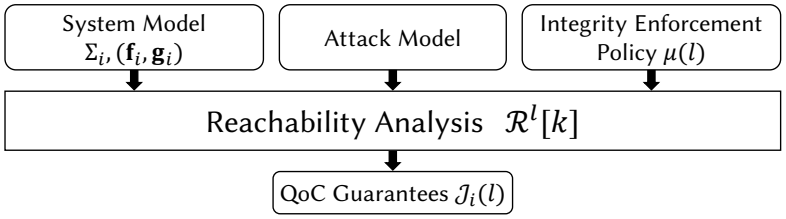


Fig. 4. Design-time framework to evaluate effects of integrity enforcement policies on QoC guarantees in the presence of attacks.

inputs. On the other hand, for each of these IDSs, a stealthy attacker cannot immediately insert any error in the state estimation; to avoid detection, the attacker rather has to craft attack signals that slowly increase estimation error. Furthermore, no actuation inputs would immediately move the system from the desired operating point due to intrinsic inertia present in all systems (which is effectively captured by the plant model Σ_i). Thus, some time has to pass after the attack starts before it significantly reduces QoC; the actual time depends on physics of the system and the compromised sensor flows.

As recently shown in [15, 16] the system (i.e., plant dynamics Σ_i and employed IDS) and attack models can be used to compute tight regions $\mathcal{R}[k]$ capturing evolution in time of the state estimation error due to any stealthy attack. Formally, the reachable region $\mathcal{R}[k]$ of the state estimation error under attack (i.e., $\mathbf{e}^a[k]$) is defined as

$$\mathcal{R}[k] = \left\{ \mathbf{e} \in \mathbb{R}^n \;\middle|\; \begin{array}{c} \mathbf{e}\mathbf{e}^\intercal \preccurlyeq E[\mathbf{e}^a[k]]E[\mathbf{e}^a[k]]^\intercal + \gamma Cov(\mathbf{e}^a_k), \\ \mathbf{e}^a[k] = \mathbf{e}^a_k(\mathbf{a}_{1..k}),\ \mathbf{a}_{1..k} \in \mathcal{A}_k \end{array} \right\},$$

where $\mathbf{a}_{1..k} = [\mathbf{a}[1]^\intercal ... \mathbf{a}[k]^\intercal]^\intercal$, $\mathcal{A}_k$ is the set of all stealthy attacks, and $\mathbf{e}^a_k(\mathbf{a}_{1..k})$ is the estimation error evolution due to the attack $\mathbf{a}_{1..k}$. Furthermore, the global reachable region $\mathcal{R}$ of the state estimation error $\mathbf{e}^a[k]$ is the set $\mathcal{R} = \bigcup_{k=0}^{\infty} \mathcal{R}[k]$. Note that here, the attack model from Section 2 can be extended with additional available information, such as the bound on the number of compromised sensor flows; if no such information is given, it is assumed that measurements from all sensors can be compromised. Furthermore, these techniques allow us to capture effects of data points in which data integrity is enforced, specified by the integrity enforcement policy μ, which can be formally defined as follows.

Definition 3.1. Intermittent data integrity enforcement policy (μ, l), where $\mu = \{t_k\}_{k=0}^{\infty}$, with $t_{k-1} < t_k$ for all $k > 0$ and $l = \sup_{k>0} t_k - t_{k-1}$, ensures that $\mathbf{a}_{t_k} = \mathbf{0}$, for all $k \geq 0$.

Note that the definition of intermittent integrity enforcement policies imposes a *maximum* time between integrity enforcements; this is the main difference compared to the standard sporadic tasks from the real-time systems literature (e.g., [8]) that can arrive at arbitrary points in time but with predefined *minimum* inter-arrival times. The definition also captures **periodic** enforcements when $l = t_k - t_{k-1}$ for all $k > 0$, and continuous integrity enforcements (with $l = 1$). Since our goal is to reduce the computation overhead associated with integrity enforcement, in this work we will focus on policies where enforcements are maximally spread apart, i.e., for which $l = t_k - t_{k-1}$ for all $k > 0$.

We argue that the evolution of state-estimation error due to attack can be used as a performance metric for QoC in the presence of attacks. This is caused by the fact that increase in the state estimation error would, through the actions of state-feedback controller, result in QoC degradation. Consequently, we obtain a design-time reachability-based framework, presented in Fig. 4. Since the

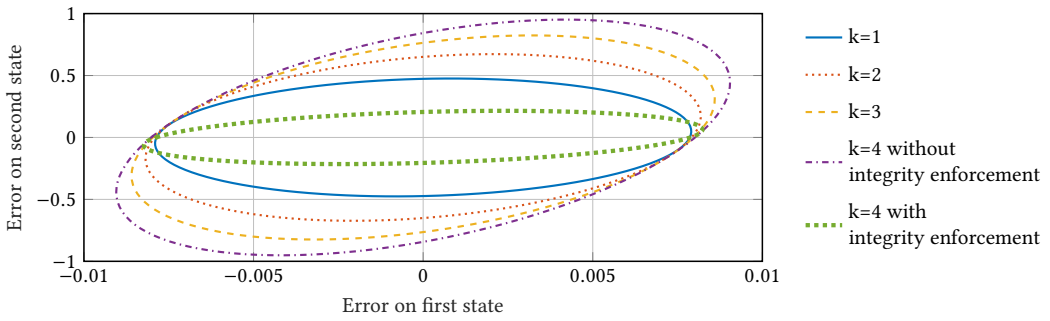


Fig. 5. Evolution of the state estimation error regions $\mathcal{R}[k]$ due to attacks, with and without data integrity enforcement at $k = 4$.

system and attack models are fixed for a control-loop under consideration, the presented framework allows for capturing the impact of the integrity enforcement parameter l on the attack-induced state estimation error (and thus QoC), through $\mathcal{J}(l)$ functions defined as

$$\mathcal{J}(l) = supp\{\|\mathbf{e}^a\|_2 \mid \mathbf{e}^a \in \mathcal{R}^l\}, \text{ where } \mathcal{R}^l = \cup_{k=0}^{\infty} \mathcal{R}^l[k].$$

Here, $\mathcal{R}^l[k]$ denotes $\mathcal{R}[k]$ computed for all integrity policies with parameter l. Functions $\mathcal{J}_i(l)$ for three automotive closed-loop systems are presented in Fig. 9. Thus, by facilitating computation of $\mathcal{J}_i(l)$ functions, the presented framework provides foundation to analyze tradeoffs between QoC guarantees in the presence of attacks and required computation resources used for data authentication. Also, QoC requirements for plant Σ_i, such as a bound on $\mathcal{J}_i(l)$, can be mapped into requirements for l_i — i.e., the number of controller invocations between consecutive data integrity enforcements.

To illustrate the effects of integrity enforcement, in Fig. 5, we show $\mathcal{R}[k]$ for a two-state vehicle model from [24], with compromised position sensor and valid velocity sensor. As can be seen, without integrity enforcements the attacker is increasing the state estimation error in each step. However, if we enforce integrity on sensor data at time $k = 4$, the estimation error significantly reduces (but does not have to go to zero).

4 MODELING OF SECURITY-AWARE CONTROL TASKS

Consider our example of two control tasks $T_1^{ctrl}(1, 4)$ and $T_2^{ctrl}(2, 4)$, and let us assume that to satisfy requirements for QoC under attack, $l_1 = l_2 = 3$ has to hold for both tasks. This results in every third task invocation, referred to as *peak*, having extended execution time by 1 time unit (as shown in Fig. 6(a), (b)). Standard real-time task model $T_i(c_i, p_i, d_i)$, where c_i is the WCET, could be used to capture these security-aware tasks. However, the effective resource utilization would be low, since the WCET of these tasks varies greatly among individual jobs. In [25] and subsequently in [4], the multiframe task model was introduced and generalized. While this model allows us to capture tasks with different execution times for different task activations, it is overly general for our current discussion, and pessimistic in the sense of task start times. Due to restrictions in the task model regarding references to any absolute time scale (in this case to the zeroth instant where the schedule begins), adding start times of jobs to the multiframe task model is non-trivial as it violates the fundamental *task independence assumption* [4]. Thus, a slightly modified approach is needed, where task execution times are allowed to differ over individual invocations given a

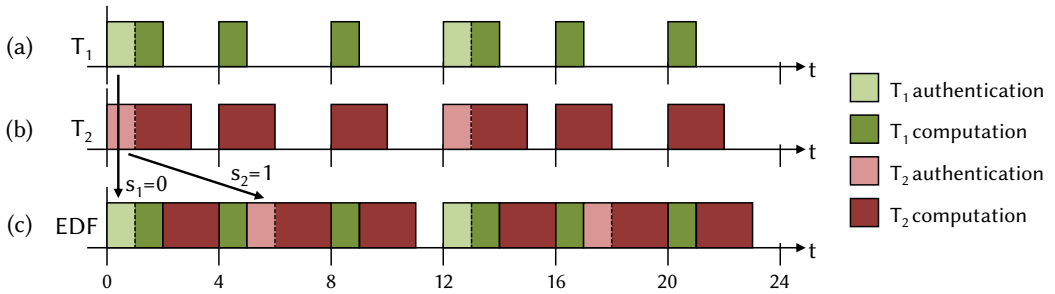


Fig. 6. Two task scheduling example: (a) and (b) show processor demand for T_1 and T_2 respectively; the task set is unschedulable if tasks were synchronous (total processor demand in the first period/frame of 4 time units is 5). However, if peak frames are asynchronous, task set is schedulable with the Earliest Deadline First (EDF) scheduler for $s_1 = 0$, $s_2 = 1$ as shown in (c).

limited pattern, but also the start time of the first peak frame[3] needs to be a controlled variable, to allow a degree of freedom during scheduling. Fig. 6(c) shows a feasible task schedule under Earliest Deadline First (EDF) scheduler when start frames s_1 and s_2 of jobs with peak execution times (peak frames) are adjusted.

Thus, we modify the multiframe task model so that the array of execution times supports exactly two parameters. The first one equals the WCET of a normal control frame (i.e., initial control task). The other is the WCET of an extended (peak) frame. Finally, we allow specification of the start time of the first peak job. Consequently, we model security-aware control tasks as a set $\mathcal{T} = \{T_1, ..., T_N\}$, where each task T_i, $1 \leq i \leq N$ is a two-frame asynchronous task defined as a 4-tuple $T_i(C_i, p_i, l_i, s_i)$ such that

- $C_i = [c_i^{ctrl}, c_i^{peak}]$ contains the WCET of two frame types, normal control and peak, characterizing task workload with and without MAC computation, respectively,
- p_i is the frame period — i.e., time between consecutive task activations,
- l_i captures inter-peak frame distance — i.e., every l_i consecutive frames contain exactly one peak frame,
- s_i is the peak frame offset that satisfies $0 \leq s_i \leq l_i - 1$, i.e., the start time of the first peak frame is $s_i p_i$.

In this task definition, we assume that deadlines of all normal and peak frames are equal to p_i, and are therefore omitted from the notation to simplify our presentation.[4] However, the approach used in this work can be easily extended to cover the general case when individual job deadlines differ from the corresponding task's period, and there could exist job activation offsets within each period (i.e., frame). This general case allows for direct capturing of effects such as worst-case network delay and jitter.

Now, for each task $T_i([c_i^{ctrl}, c_i^{peak}], p_i, l_i, s_i)$ and the task set $\mathcal{T}$, we define the task (U_i) and task-set ($U_{\mathcal{T}}$) utilizations, respectively, as

$$U_i = \frac{c_i^{ctrl}}{p_i} + \frac{c_i^{peak} - c_i^{ctrl}}{l_i p_i}, \qquad U_{\mathcal{T}} = \sum_{i=1}^{N} U_i.$$

[3]We will refer to task invocations (or jobs) as frames, similarly as proposed in [4, 25].
[4]Note that the two-frame task with peak frame offset provides the same modeling expressiveness as the composition of two standard tasks one of which has offset $s_i p_i$ and period $l_i p_i$, as long as precedence constraints are established among them.

This definition allows us to obtain a schedulability condition that can be simply verified; the lemma's proof directly follows from [25].

Lemma 4.1. *The task set $\mathcal{T}$ is not schedulable under any scheduling policy if $U_{\mathcal{T}} > 1$.*

Recall our example task set consisting of two tasks (see Fig. 6). These tasks can now be specified as: $T_1([1,2],4,3,s_1), T_2([2,3],4,3,s_2)$. Although $U_{\mathcal{T}} = 0.92$, this task set is not schedulable under EDF for some values of start times s_i. However, for example, for $s_1 = 0$ and $s_2 = 1$, EDF can schedule the tasks as illustrated in Fig. 6(c). Thus, the presented task set $\mathcal{T}$ is incomplete, in the sense that peak frame offsets $s_1, ..., s_N$ are not specified. Given such an incomplete task set, first and foremost we are interested in determining a set of peak frame offsets that makes this task set schedulable under EDF, if that is at all possible; we consider EDF as it is known to be optimal non-idle scheduler. Secondly, we wish to maximize utilization of available resources for feasible task sets, i.e., maximize the overall QoC under attack while ensuring that the task set is still schedulable. As described in Section 3, since for each control loop i, degradation of QoC in the presence of attacks can be captured as a function of the times between consecutive integrity enforcements l_i (i.e., $\mathcal{J}_i(l_i)$), we specify the overall QoC degradation as $\sum_{i=1}^{N} \omega_i \mathcal{J}_i(l_i)$ for some positive weights ω_i, $i = 1, ..., N$, which are used to 'emphasize' QoC for some tasks compared to others.

Therefore, we can formally define the two problems as follows:

Problem 1. *For a task set $\mathcal{T}$ with $l_1, ..., l_N$ capturing prespecified QoC requirements, find* feasible *peak frame offsets $s_1, ..., s_N$ such that the obtained task set $\mathcal{T}$ is schedulable under EDF.*

Problem 2. *For a task set $\mathcal{T}$ and a set of associated cost functions $\mathcal{J}_i(l_i)$, $i = 1, ..., N$, find peak frames' offset values $s_1, ..., s_N$ and optimal peak frame periods $l_1, ..., l_N$ such that the resulting task set $\mathcal{T}$ is schedulable under EDF and objective $\sum_{i=1}^{N} \omega_i \mathcal{J}_i(l_i)$ is minimized.*

5 SCHEDULING WITH QOC REQUIREMENTS

In this section, we provide a method to find a set of feasible peak frame offsets $s_1, ..., s_N$ based on the *processor demand criterion* [5, 8].

Definition 5.1. [5] The demand function df_i of a standard task T_i on an interval $[t_1, t_2]$ is

$$df_i(t_1, t_2) = \sum_{a_{i,j} \geq t_1,\ d_{i,j} \leq t_2} c_i, \tag{1}$$

where c_i is the WCET of the i^{th} task, while $a_{i,j}$ represents time of the j^{th} job arrival, and $d_{i,j}$ its respective deadline.

In other words, the demand function is equal to the sum of processor demand for all jobs of the task that have both their activation time and deadline in the time period $[t_1, t_2]$. Intuitively, it quantifies the amount of work the processor will be presented with during the interval $[t_1, t_2]$. For example, the demand function $df_1(0, t)$ for task T_1 from our running example, with $s_1 = 0$, is shown in Fig. 7.

The following theorem formulates the necessary and sufficient condition for feasibility of asynchronous task set.

Theorem 5.2. *[5] A task set $\mathcal{T}$ is schedulable by EDF if and only if $\sum_i df_i(t_1, t_2) \leq t_2 - t_1$, for all t_1, t_2 such that $t_1 < t_2$.*

Note that the condition from the above theorem can be significantly simplified for synchronous tasks (for which start times of all tasks are fixed to the zeroth instant) with a feasibility test based

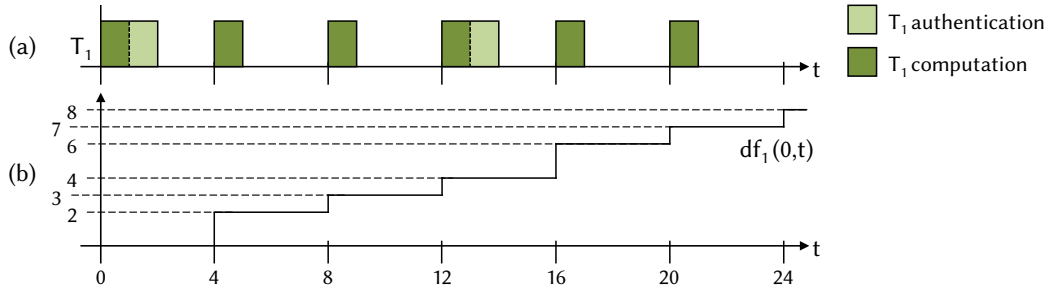


Fig. 7. Demand function example: (a) Graphical representation of a two-frame task $T_1([1,2],4,3,0)$, and (b) a demand function $df_1(0,t)$ for task T_1, as defined in (1) for varying t and start (t_1) at 0.

on the demand bound function $dbf_i(t) = \max_t df_i(t', t'+t) = df_i(0,t), \forall t' \geq 0$. However, this does not apply for our task model due to the asynchronicity involved in the peak frame start time.

Given the piecewise constant nature of the demand function and task periodicity, the schedulability condition from Theorem 5.2 has to be evaluated only in a finite number of points corresponding to task arrivals and deadlines [8]. We will refer to this set of time instants as the *time-testing set*. Given that absolute deadlines of jobs in our task set are always exactly p_i away from their activation, the time-testing set can be obtained as

$$TS = \bigcup_{i=1}^{N} \{\, t \mid t = kp_i \wedge t \leq P_H \wedge k \in \mathbb{N}_0 \},$$

where $P_H = lcm\{l_1 \cdot p_1, l_2 \cdot p_2, ..., l_N \cdot p_N\}$ is the hyperperiod of the schedule and $lcm(.)$ is the least common multiple. For our running example (see Fig. 6(c) for schedule), hyperperiod and the time testing set are $P_H = 12$, $TS = \{0, 4, 8, 12\}$. It is worth noting that we only have to check up to P_H since our start times are constrained — i.e., absolute start time of the peak frame may not exceed $s_i p_i$. Otherwise, for general start times, the result from [20] applies — the time bound up to which the processor demand test should be conducted is $\max_i \phi_i + 2P_H$, where ϕ_i are absolute task start times.

To devise the analytical expression for the demand function, we obtain the number of normal control and peak jobs of our task activated and required to complete in an interval $[t_{k_1}, t_{k_2}]$ as

$$\eta_i^{c\&p}(t_{k_1}, t_{k_2}) = max\left\{0, \left\lfloor \frac{t_{k_2} - p_i}{p_i} \right\rfloor - max\left\{0, \left\lceil \frac{t_{k_1}}{p_i} \right\rceil\right\} + 1\right\}.$$

Similarly, it can be calculated that the number of peak jobs contributing to the demand during the same interval is

$$\eta_i^{peak}(t_{k_1}, t_{k_2}) = max\left\{0, \left\lfloor \frac{t_{k_2} - (s_i+1)p_i}{l_i p_i} \right\rfloor - max\left\{0, \left\lceil \frac{t_{k_1} - s_i p_i}{l_i p_i} \right\rceil\right\} + 1\right\}.$$

Therefore, the processor demand function over $[t_{k_1}, t_{k_2}]$ can be compactly captured as

$$df_i(t_{k_1}, t_{k_2}) = c_i^{ctrl} \eta_i^{c\&p}(t_{k_1}, t_{k_2}) + \Delta c_i \eta_i^{peak}(t_{k_1}, t_{k_2}),$$

where $\Delta c_i = (c_i^{peak} - c_i^{ctrl})$. Finally, for a task set $\mathcal{T}$ it follows from Theorem 5.2 and the discreteness of the time testing set TS that a necessary and sufficient feasibility test can be formulated as

$$\sum_{i=1}^{N} df_i(t_{k_1}, t_{k_2}) \leq t_{k_2} - t_{k_1}, \quad \forall t_{k_1}, t_{k_2} \in TS,\ t_{k_1} < t_{k_2}. \tag{2}$$

In the following subsection we analyze the transformation of the demand function to a set of linear constraints, which will provide basis for our MILP-based scheduling with QoC requirements.

5.1 Mixed Integer Linear Programming Formulation for Schedule Synthesis

In Problem 1, our goal is to derive feasible peak frame offsets for all tasks in the task set such that none of the deadlines are missed when the EDF scheduler is used. Note that at every instant from the time testing set TS, the demand function df_i is a function of the start time s_i since all other values are specified. Hence, the problem of testing feasibility of a task set under dynamic-priority scheduling policies can be directly mapped into an MILP as follows.

Let $a^i_{k,j}$ be binary variables indicating that by the k^{th} instant from the time testing set, the j-th peak frame of task T_i has been scheduled to complete execution, where $1 \le i \le N$, $1 \le j \le \frac{P_H}{l_i p_i}$, $2 \le k \le |TS|$; note that here $k \ge 2$, since it must hold that $a^i_{1,j} = 0$, and thus it is not necessary to have a set of variables at the first point of time-testing set (i.e., for $k = 1$). The dependency of variables $a^i_{k,j}$ to the task parameters can be formally captured as

$$a^i_{k,j} = 1 \Leftrightarrow t_k \ge (s_i + 1)p_i + (j-1)l_i p_i. \tag{3}$$

For instance, for the schedule shown in Fig. 6(c), corresponding variables are $a^1_{2,1} = 1, a^1_{3,1} = 1, a^1_{4,1} = 1$ for T_1 and $a^2_{2,1} = 0, a^2_{3,1} = 1, a^1_{4,2} = 1$ for task T_2, given that peak frame offsets were chosen as $s_1 = 0$, $s_1 = 1$. Hence, from the definition of $\eta_i^{peak}(t_{k_1}, t_{k_2})$ it follows that $\eta_i^{peak}(t_{k_1}, t_{k_2}) = \sum_{j=1}^{\frac{P_H}{l_i p_i}} \left(a^i_{k_2,j} - a^i_{k_1,j}\right)$. This implies that for any t_{k_1} and t_{k_2} ($t_{k_2} \ge t_{k_1}$) from the time-testing set TS, the demand function for task T_i can be expressed as a function of only binary variables $a^i_{k,j}$ in the form

$$df_i(t_{k_1}, t_{k_2}) = c_i^{ctrl}\eta_i^{c\&p}(t_{k_1}, t_{k_2}) + \Delta c_i \sum_{j=1}^{\frac{P_H}{l_i p_i}} \left(a^i_{k_2,j} - a^i_{k_1,j}\right). \tag{4}$$

Consequently, there exist feasible peak frame offsets $s_1, \ldots, s_N$, such that task set $\mathcal{T}$ is schedulable with EDF, if and only if the following set of mixed-integer linear constraints has a feasible solution

$$\sum_{i=1}^{N} df_i(t_{k_1}, t_{k_2}) \le t_{k_2} - t_{k_1}, \quad \forall t_{k_2}, t_{k_1} \in TS \tag{5}$$

$$(s_i + 1)p_i + (j-1)l_i p_i \le t_k + M(1 - a^i_{k,j}) \tag{6}$$

$$(s_i + 1)p_i + (j-1)l_i p_i > t_k - M a^i_{k,j} \tag{7}$$

$$0 \le s_i \le l_i - 1. \tag{8}$$

Here, df_i is defined as in (4), the variable indices satisfy

$$1 \le i \le N, \qquad 1 \le j \le \frac{P_H}{l_i p_i}, \qquad 2 \le k \le |TS|,$$

and M is a large constant. Constraints (6) and (7) capture the logical condition specified in (3) using the standard "Big M" method to capture indicator constraints [6].

Remark 1. *Most available MILP solvers require the set of constraints to be specified as non-strict inequalities (i.e., in the form* $\mathbf{Ax} \le \mathbf{b}$*). Thus, we can express constraint (7) as a non-strict inequality by adding a small* $\varepsilon > 0$ *to every* t_k *– i.e., as*

$$(s_i + 1)p_i + (j-1)l_i p_i \ge t_k - M a^i_{k,j} + \varepsilon \tag{9}$$

Yet, M and ε have to be carefully chosen to avoid potential errors due to finite precision implementations of MILP solvers. Consider that (9) expresses peak frame timing conditions when $a^i_{k,j} = 0$. Thus, M and ε should be chosen so that the following inequalities are not violated

$$M\delta_{int} + \delta_{constr} < \varepsilon < 1 - M\delta_{int} - \delta_{constr},$$

where δ_{int} and δ_{constr} are tolerances of MILP solvers. Specifically, δ_{int} is the maximum deviation from an integer value a variable can have while still being considered an integer, and δ_{constr} is the maximum discrepancy that can be involved in a linear constraint while it is still being considered satisfied. Similarly, M has to be set sufficiently large so that for large t_k-s from TS, (6)-(7) are still satisfied when tolerances are taken into account.

Note that we do not specify an objective function since solely feasibility is of interest here, i.e., we want to determine *feasible peak frame offsets*. If the given task set is schedulable, solving this MILP problem will result in a concrete set of values for $s_1, ..., s_N$, which complete our task set. If the feasible set of the MILP problem is empty, the task set under consideration is not feasible.

The schedulability constraints specified in (4)-(8) feature $(|TS| - 1)\sum_{i=1}^{N} \frac{P_H}{l_i p_i}$ binary (i.e., $a^i_{k,j}$) and N integer variables (i.e., s_i). Furthermore, (5) results in $\binom{|TS|}{2}$ constraints, while (6) and (7) each add $|TS|\sum_{i=1}^{N} \frac{P_H}{l_i p_i}$ constraints. For our simple running example, there are 6 binary and 2 integer variables, and a total of 18 constraints (not including variable bound constraints). As shown in Sec. 7, the MILP problem's size does not impose stringent limitations for realistic systems and workloads.

6 SYNTHESIS OF QOC-OPTIMAL SCHEDULES

The previous section presents a set of mixed-integer linear constraints that specify necessary and sufficient schedulability conditions for predefined QoC requirements, which are captured as the values of l_i task parameters. On the other hand, to optimally use available resources, overall QoC guarantees can be improved by increasing the rate of integrity enforcements for underutilized systems or by decreasing QoC guarantees if the initial task set is infeasible. Thus, in this section we present a MILP-based approach to solve Problem 2 which requires minimization of the overall QoC degradation in the presence of attacks — i.e., $\sum_{i=1}^{N} \omega_i \mathcal{J}_i(l_i)$ objective.

We start by noting that the set of schedulability constraints remains linear if $l_1, ..., l_N$ become variables, instead of predefined QoC parameters. Still, several challenges need to be addressed. First, the time-testing set and thus the number of binary variables depend on the size of hyperperiod P_H and hence the values of l_i, which in this case are variables. Since we assume that for each control task, minimal QoC under attack requirements are specified in the form of the maximal allowed inter-peak frame distance l_i^{max}, it is possible to provide an upper bound on the size of hyperperiod

$$\overline{P}_H = \max_{\substack{l_i \in \{1,...,l_i^{max}\} \\ i \in \{1,...,N\}}} lcm(l_1 p_1, ..., l_N p_N),$$

which can be precomputed. Note that this could potentially result in a larger than necessary time-testing set TS, but would still guarantee schedulability if conditions from (2) are satisfied.

The second challenge is that the functions $\mathcal{J}_i(l_i)$ are obtained only through the use of the presented reachability framework from Fig. 4 and thus no analytic solutions are available in the general case. To address this, we start by noting that for realistic control systems the aforementioned functions $\mathcal{J}_i(l_i)$ can be well-approximated using piecewise linear functions $\hat{\mathcal{J}}_i(l_i)$. Examples of these cost functions and their approximations are shown in Fig. 9. Let F_i denote the number of linear segments used to approximate the experimentally obtained QoC degradation function $\mathcal{J}_i(l_i)$ for the i^{th} task. For example, for the cost function shown in Fig. 9(a) there are $F_1 = 2$ segments:

$[\ \underline{l}_1^1 = 1, \bar{l}_1^1 = 2\ ]$ and $[\ \underline{l}_1^2 = 2, \bar{l}_1^2 = 5\ ]$, respectively. Then, the QoC degradation function can be approximated as

$$\hat{\mathcal{J}}_i(l_i) = \sum_{r=1}^{F_i} \left((\alpha_r^i l_i + \beta_r^i) b_r^i \right), \tag{10}$$

where $\alpha_r^i l_i + \beta_r^i$ is the r^{th} linear segment $[\ \underline{l}_i^r, \bar{l}_i^r\ ]$, out of F_i, such that $[1, l_i^{max}] = \bigcup_{r=1}^{F_i} [\ \underline{l}_i^r, \bar{l}_i^r\ ]$, and b_r^i are binary variables that select the corresponding linear segment based on the value of l_i. This is captured through a set of logical conditions

$$b_r^i = 1 \qquad \Rightarrow \qquad \underline{l}_i^r \le l_i \le \bar{l}_i^r, \quad 1 \le r \le F_i.$$

Similarly as before, using the "Big M" method, the above logical condition can be specified as a set of the following linear constraints

$$\begin{aligned} \underline{l}_i^r - M(1 - b_r^i) \le l_i \le \underline{l}_i^r + M b_r^i, \\ \bar{l}_i^r - M b_r^i \le l_i \le \bar{l}_i^r + M(1 - b_r^i), \end{aligned} \tag{11}$$

where M is a large constant. Additionally, a constraint limiting that only one linear segment is active per task is needed — i.e.,

$$\sum_{r=1}^{F_i} b_r^i = 1, \qquad 1 \le i \le N. \tag{12}$$

Finally, in the objective function in (10), the only nonlinearities are the multiplications of variables $l_i b_r^i$. These can be linearized by introducing a new set of variables $c_r^i = l_i b_r^i$, which can be expressed using the "Big M" method as the following linear constraints

$$c_r^i \le b_r^i M, \qquad c_r^i \ge l_i - (1 - b_r^i) M, \qquad 0 \le c_r^i \le l_i. \tag{13}$$

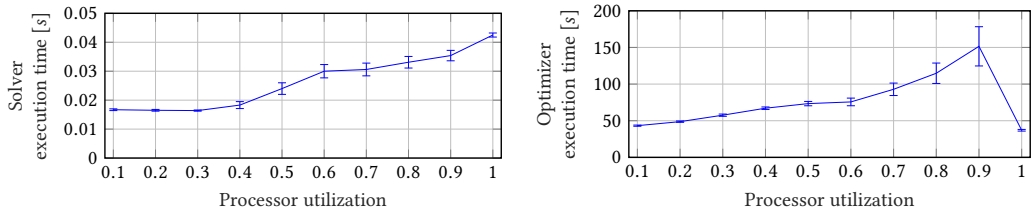
The first constraint in (13) guarantees that $c_r^i = 0$ when $b_r^i = 0$, i.e., when the r^{th} segment is not selected. The second constraint sets $c_r^i = l_i$ when corresponding segment is selected, i.e., when $b_r^i = 1$. Therefore, the complete MILP formulation for QoC-optimal task set synthesis can be specified as

$$\begin{aligned} &\min \sum_{i=1}^{N} \left[\omega_i \sum_{r=1}^{F_i} (\alpha_r^i c_r^i + \beta_r^i b_r^i) \right] \\ &\text{subject to: (4)-(8), (11)-(13)} \\ &1 \le i \le N, \qquad 1 \le j_i \le \frac{\overline{P}_H}{p_i}, \qquad 2 \le k \le |TS|. \end{aligned} \tag{14}$$

In addition to $(|TS| - 1) \sum_{i=1}^{N} \frac{\overline{P}_H}{p_i}$ binary (a_{k,j_i}^i-s) and N integer variables (s_i-s), this MILP formulation adds $N + \sum_{i=1}^{N} F_i$ integer variables (l_i-s and c_r^i-s), and $\sum_{i=1}^{N} F_i$ binary variables (cost function segment selector variables b_r^i-s). In addition to $\binom{|TS|}{2} + 2 \cdot |TS| \sum_{i=1}^{N} \frac{\overline{P}_H}{p_i}$ constraints that exist in the schedulable task set synthesis formulation, N restrictions for single linear segment selection (expressed in (12)) and $4 \cdot \sum_{i=1}^{N} F_i$ constraints covering (11) and (13) that constrain linearization variables (c_r^i-s) have to be added. For our running example, there are 76 binary and 8 integer variables, and a total of 185 constraints, if two-segment linear curves are combined to form an objective function. Trivial variable bound constraints are not included in this count.

Finally, let us revisit our introductory two-task example:

$$T_1([1, 2], 4, 3, s_1), \qquad T_2([2, 3], 4, 3, s_2). \tag{15}$$



(a) Schedule synthesis with predefined QoC requirements

(b) QoC-optimal schedule synthesis

Fig. 8. Execution times of Gurobi MILP solver with 95% confidence intervals for the task set in (16).

As we mentioned in previous sections, this task set is feasible under EDF for peak frame offset assignment $s_1 = 0, s_2 = 1$. Consider the case when, for example, integrity enforcement for the process loop controlled by task T_1 is required every five periods instead of three, (i.e., $l_1 = 5$). If schedulable task set synthesis formulation from Section 5.1 is applied, even though effective processor utilization is lower, this task set becomes infeasible for any start time assignment. This is indeed the case since peak frame periods are such that, under any assignment of peak frame start times, peak frames of the two tasks will eventually align and cause the processor demand test to fail (processor demand over a period of 4 time units will be 5). This is due to the fact that the formulation specified in (5)-(8) only tests feasibility of the given task set and determines feasible peak frame offsets if possible. However, if QoC-optimal scheduling formulation, as specified in (14), is applied for $l_1^{max} = 3, l_2^{max} = 5$, the result of optimization will be a feasible and optimal assignment for variables l_i and s_i, that minimizes the QoC objective. In this case, $l_1 = l_2 = 2$ and $s_1 = 0, s_2 = 1$ is the output of the QoC-optimization, if standard QoC degradation functions are used as objectives.

7 EVALUATION

To analyze scalability and performance of our approach, we evaluate the proposed framework both on random workloads and a realistic automotive case-study.

7.1 General Evaluation and Limitations

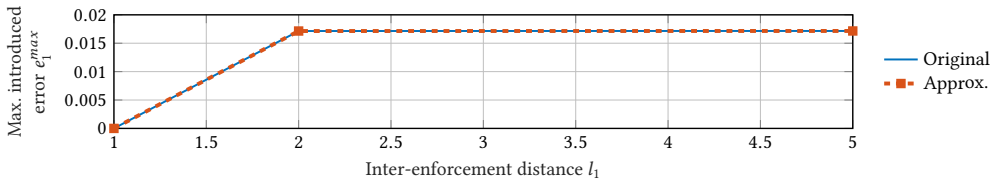
We start our evaluation by considering execution times of Gurobi MILP solver [26] for generic task sets. All execution times are measured on a platform with a 5^{th} generation $3.0GHz$ Intel i7 CPU and $16GB$ of memory.

We construct generic task sets by varying overall processor utilization, and randomly generating workloads according to the current utilization set point, while keeping the task periods fixed. It is important to highlight that task sets that are constructed fully randomly do not give us full insight about performance of our framework; this is caused by the fact that the size of our MILP formulation is dominated by the size of the time-testing set ($|TS|$) and the number of tasks' peak frames during the scheduling hyperperiod ($\sum_{i=1}^{N} \frac{P_H}{l_i p_i}$), and is less affected by the number of tasks.

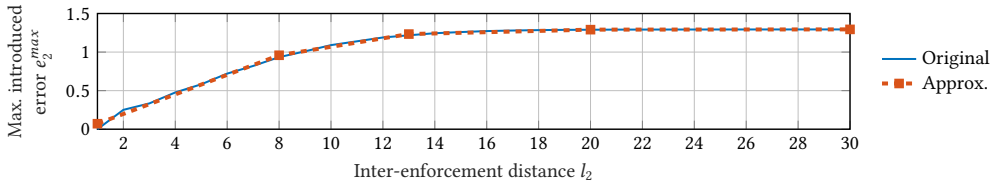
To illustrate this, consider a task set consisting of four tasks

$$T_1([0.1, 0.2], 4, 4, s_1), \qquad T_2([0.1, 0.3], 4, 4, s_2),$$
$$T_3([0.1, 0.4], 12, 6, s_3), \qquad T_4([0.1, 0.3], 12, 2, s_4).$$

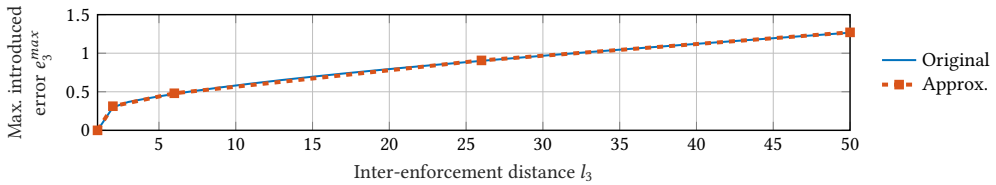
Due to relatively low processor demand with $U < 0.1$, this task set is schedulable. The MILP formulation features a total of 936 binary and 4 integer variables, while cardinality of the time



(a) Fuel injection



(b) Driveline management



(c) Trajectory tracking

Fig. 9. Change in the maximal state estimation error with respect to integrity enforcement period for fuel injection, driveline management and trajectory tracking systems. The estimation error for fuel injection increases quickly, making integrity enforcement at all times inevitable, while state estimation errors for driveline management and trajectory tracking systems reach unsafe levels for $l_2 > 10$ and $l_3 > 5$, respectively. Finally, without integrity enforcements, the trajectory tracking system is a *perfectly attackable system* [18, 24], meaning that the attacker can introduce unbounded estimation error.

testing set is 37. There is a total of 2538 constraints (not including variable bound constraints). Note that task periods p_i-s and the number of frames between two consecutive peak frames l_i-s are valued such that the size of the problem is relatively small. To obtain a schedulable task-set, Gurobi MILP solver takes 8.3 *ms*. However, if task periods are such that the processor demand condition (5) requires evaluation in a larger number of time instants, this results in a significant increase in the problem size. This occurs when periods and peak frame distances are not harmonically related. For instance, if T_3's periodicity is 13 time units instead of 12, the total number of variables becomes 21508, while the time testing set's size is 193. The total number of constraints is 61540, and Gurobi execution time increases to 535.4 *ms*.

On the other hand, in most control applications, periods of control tasks are multiples of a small set of numbers (e.g., 10 *ms* and 20 *ms* in most automotive control applications), which enables construction of reasonably sized problems that can be efficiently solved. Additionally, the described interleaving of peak frames that is inevitable for certain combinations of task parameters (illustrated with example in (15)) is much less likely to occur in real systems where the integrity enforcement policy can be adjusted to avoid this. As a result, most realistic task sets can be efficiently scheduled using our approach.

Table 1. Automotive case-study task set with fuel injection (T_1), driveline management (T_2), and trajectory tracking (T_3) tasks, as well as additional logging and supervision tasks; Shaded columns are results of QoC-optimal schedule synthesis. Light shaded columns show optimization results for $\mathcal{T}_1 = \{T_1, ..., T_6\}$ ($U_{\mathcal{T}_1} = 0.82$); dark shaded cells mark additional tasks and show optimization results for $\mathcal{T}_2 = \{T_1, ..., T_8\}$ ($U_{\mathcal{T}_2} = 0.87$).

Task	p_i $[ms]$	c_i^{ctrl} $[ms]$	c_i^{peak} $[ms]$	l_i^{max}	s_i^*	l_i^*	s_i^*	l_i^*
T_1	10	1.6068	2.7012	1	0	1	0	1
T_2	20	2.6172	5.3346	10	0	2	0	2
T_3	20	1.6068	2.7012	5	1	2	1	4
T_4	100	1.8258	–	–	–	–	–	–
T_5	100	7.8652	–	–	–	–	–	–
T_6	200	5.5587	–	–	–	–	–	–
T_7	40	1.4427	–	–	–	–	–	–
T_8	50	0.8562	–	–	–	–	–	–

Fig. 8 presents execution times of Gurobi MILP solver for both schedulable task set synthesis with predefined QoC requirements and QoC-optimal schedule synthesis. We analyzed how Gurobi execution times depend on utilization for task set

$$\begin{aligned} &T_1([c_1^{ctrl}, c_1^{peak}], p_1 = 10, l_1^{(max)} = 1, s_1), \quad && T_2([c_2^{ctrl}, c_2^{peak}], p_2 = 20, l_2^{(max)} = 8, s_2), \\ &T_3([c_3^{ctrl}, c_3^{peak}], p_3 = 40, l_3^{(max)} = 4, s_3), \quad && T_4([c_4^{ctrl}, c_4^{peak}], p_4 = 120, l_4^{(max)} = 2, s_4). \end{aligned} \tag{16}$$

Here, for synthesis of schedulable task sets with predefined QoC requirements we use $l_i = l_i^{max}$ from (16). While randomly generating task sets, normal and peak frame execution times were chosen randomly 100 times so that utilization of a task is proportional to its period. Intuitively, for higher utilizations, the optimizer explores a larger space in search for a feasible solution. In the limit, a task set with utilization $U = 1$ is very unlikely to be feasible. Hence, an MILP solver implementing branch and bound algorithm can easily prune out large portions of the variable space, resulting in a decrease in execution time as shown in Fig. 8(b). Note that for schedule synthesis (Fig. 8(a)), the variable space is relatively small and thus the solver execution time is very low (57.3 *ms* for worst run).

7.2 Automotive Case Study

We demonstrate the usability of our approach on a realistic case study involving three automotive control components. Here, we consider control tasks for fuel injection, driveline management, and trajectory tracking (additional use of intermittent integrity enforcements on different platforms, e.g., vehicle-to-vehicle/infrastructure (V2V/I) can be found in [15]). Modeling methodology for these systems is thoroughly described in [31], [30], and [17], respectively. As described in Section 3, we use the models of these systems to quantify QoC degradation in the presence of attacks using cost functions $\mathcal{J}_i(l_i)$. These functions along with their piecewise-linear approximations ($\hat{\mathcal{J}}_i(l_i)$ as captured by (10)) are shown in Fig. 9.

We abstract the computational workload of controlling these systems with tasks T_1,T_2, and T_3, respectively. Task set parameters are given in Table 1. For the fuel injection system, we observe that relatively small error in state estimation results in significant changes in air-to-fuel ratio (AFR), which is one of the main controlled states in this system. Small changes in AFR can have

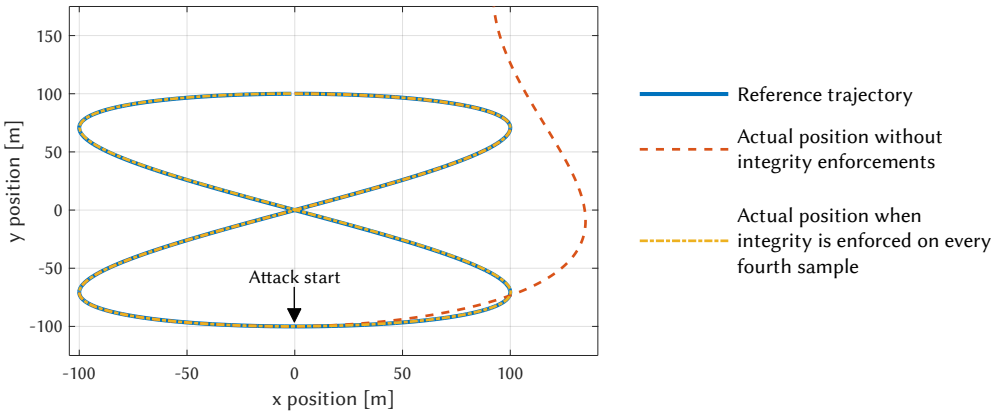


Fig. 10. Reference trajectory and obtained position under attack with and without integrity enforcement are shown. Simulation duration is 200 s and the attack starts at 100 s.

significant impact on engine performance (increased emissions, poor power output, and overheating). Therefore, we choose to enforce integrity on every data point (i.e., $l_1^{max} = 1$). The driveline management system features drive-shaft torsion, engine speed, and wheel speed as its states. Given that significant change in torsion leads to increased wear of mechanical components of driveline and potential permanent damage, we choose to limit the maximum allowed state estimation error to 0.02 rad (or 1.15°) for drive-shaft torsion. Since engine speed is typically significantly larger than wheel speed, we set the limits on the remaining states' estimation errors to 1 $\frac{rad}{s}$ and 0.25 $\frac{rad}{s}$ respectively. These constraints result in maximal allowed state estimation error of $e_2^{max} = 1.12$. By mapping this value through the QoC degradation function in Fig. 9(b), we obtain the maximum inter-enforcement period of $l_2^{max} = 10$ control periods. For trajectory tracking, we allow additional error induced by the attacker to be very small, precisely no more than 0.35 $\frac{m}{s}$ in estimated speed, and 0.3 m in position. This gives us the maximum state estimation error of $e_3^{max} = 0.461$. Mapping through Fig. 9(c) gives us the maximum sample distance between two integrity enforcements of $l_3^{max} = 5$.

To capture realistic scenarios where ECUs are shared between control- and non-control tasks, we also add tasks T_4-T_6 specified in Table 1 as standard real time tasks which have no QoC degradation function associated with them. These tasks execute additional logging and supervision functions (e.g., gearbox oil temperature checking and logging with period 200 ms). Notice that no changes in our formulation are necessary to allow admission of such tasks (simple declaration $c_i^{peak} = 0$, $l_i = 1$ suffices). Utilization for task set $\mathcal{T}_1 = \{T_1, ..., T_6\}$ is $U_{\mathcal{T}_1} = 0.82$.

For the aforementioned task set, Gurobi takes 3.9 ms to find a feasible set of peak frame offsets $s_1, ..., s_6$. For QoC-optimal task synthesis the MILP solver's execution time is 764.3 s, and the results are shown in the light shaded columns of Table 1 (optimal s_i^* and l_i^*). For these values, to illustrate QoC under attack, we simulate the vehicle motion with a figure eight-shaped reference trajectory that fits inside a 100×100 m square. Simulation duration is 200 s, and attack start time is set to 100 s. Fig. 10 shows the difference between reference and obtained vehicle trajectories during simulation time. As can be seen, integrity enforcement prevents the attacker from significantly diverting the system away from the desired trajectory.

To examine system scalability and evaluate performance degradation due to increase in task set utilization, we add two more tasks to our task set, namely T_7 and T_8. Task parameters and the result of the new optimization problem based on the extended task set $\mathcal{T}_2 = \{T_1, ..., T_8\}$ are shown

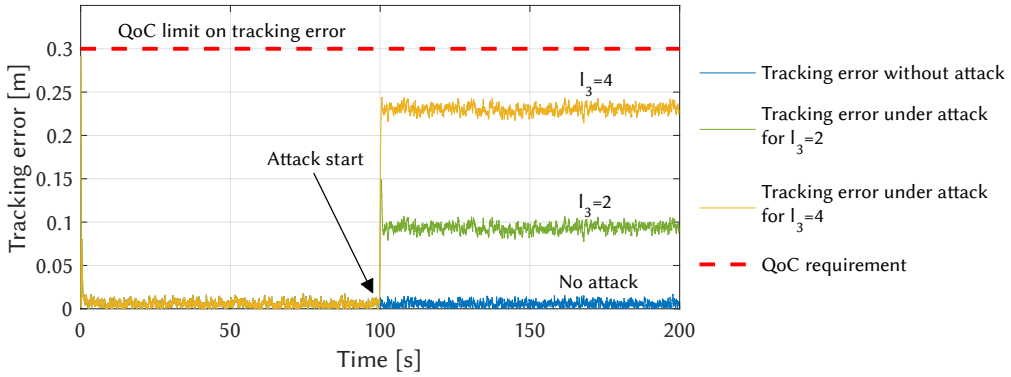


Fig. 11. Tracking error over the course of the simulation. The error is presented in the case without and with the presence of attack for $l_3 = 2$, and $l_3 = 4$. Note that design-time performance requirements are satisfied in all cases.

in darker shaded cells of Table 1. Utilization for this task set is $U_{\mathcal{T}_2} = 0.87$. Schedule synthesis with predefined QoC requirements for this task set takes 4.2 *ms* to execute, while Gurobi execution time for QoC-optimal task synthesis is 729.6 *s*. As a result of higher processor utilization and specific task periods, task T_3 authenticate sensor data only as often as every fourth control period, instead of every other.

A typical QoC metric for trajectory tracking systems is the tracking error, i.e., the difference between the obtained and desired trajectories. As can be seen from Fig. 11, the decrease in integrity enforcement rate increases potential influence of the attacker. Nevertheless, the tracking error is still maintained under design requirements.

Finally, when we increase the utilization up to 0.9936 by adding more tasks, we observer that the values for l_1, l_2 and l_3 obtained from the optimization are equal to design-time limits, i.e., $l_1^* = 1$, $l_2^* = 10$, $l_3^* = 5$, meaning that even with this utilization our task set is still schedulable while QoC requirements are satisfied. It is worth noting that in this case data integrity is enforced in only 15% of time steps for controllers implemented in tasks T_2 and T_3, allowing for execution of additional tasks on the shared CPU.

8 RELATED WORK

Security challenges due to the tight interaction of cyber and physical components in CPS have attracted a lot of attention in recent years. One focus has been on the use of control theory to develop attack-resilient algorithms and architectures (e.g., see a recent study [23] and references therein). While some of existing works consider implementation issues such as jitter [28], to the best of our knowledge, this work is the first to address closed-loop performance (i.e., QoC), data integrity, and schedulability guarantees within an integrated resource-aware design and analysis framework.

Intermittent use of integrity enforcements for embedded control systems is similar in spirit to event- and self-triggered control [1, 2] control. In addition, adding security mechanisms to resource constrained embedded and CPS, and effects on real-time scheduling of existing tasks was addressed in [14, 22, 32]. In [14], opportunistic execution of security tasks in legacy systems is proposed, by optimizing their execution parameters while ensuring feasibility of existing tasks. Also, [32] presents a scheduling policy that takes into account security requirements together with real-time requirements for embedded systems. In [22], active security services are optimally chosen with

respect to schedulability conditions, while simultaneously guaranteeing schedulability through modified conditions for EDF. Yet, in all these works, security guarantees are captured with the use of abstract *security levels*; there is no connection between the available resources and performance of the main system functionalities (e.g., QoC for control tasks) in the presence of attacks. On the other hand, the proposed framework based on intermittent integrity enforcements facilitates tradeoff analysis between system resources and QoC guarantees.

Some of these problems can be framed as task adaptation in overutilized systems. In [9], an elastic task model is presented that adapts Quality-of-Service (QoS) in control applications where relaxation of timing constraints is allowed. Similarly, QoC captured as a standard quadratic control cost, is used in [10] for adaptation of task periods in order to track a predefined utilization setpoint. These methods cannot be directly applied to adaptation of multiframe security-aware real-time control tasks, since they are based on the standard task model, for which they alter control tasks' periods, while we assume that periods of existing control tasks cannot be changed and focus on optimal inclusion of integrity enforcements.

Finally, MILP is used in real-time systems for multiprocessor sporadic task partitioning [3] and to find feasible deadlines under EDF for a given performance index [7]. In [29], task parameter adaption for multiframe tasks is solved via MILP; still, only the worst case alignment of frames is considered, and unlike our approach to finding feasible offsets, the focus is on period and deadline optimization.

9 CONCLUSION

We have presented a method to add security guarantees, in terms of Quality-of-Control in the presence of attacks on sensor measurements, to control tasks executed on a shared processor. We have exploited the fact that even intermittent data integrity enforcements significantly limit the attacker's impact on the system and shown how to obtain the relationship between the integrity enforcement rate and QoC under attack. This has allowed us to map the problem into scheduling of security-aware multiframe tasks, for which we have presented an MILP formulation. Furthermore, we have introduced a MILP-based approach for optimal resource (i.e., CPU time) allocation, which allows for the maximization of the overall QoC while ensuring task system schedulability. Finally, the usability of our approach has been illustrated on an automotive case study.

Note that the proposed approach results in periodic data integrity enforcements, though with a significantly reduced rate and thus significantly reduced overhead. For instance, we have shown that we can have satisfiable QoC under attack with as low as 15% of controller executions for which data integrity is guaranteed. An avenue for future work is the use of fully intermittent integrity enforcement policies, which could potentially allow for QoC improvements for desired utilization levels. We will also combine the presented framework with our recent work on scheduling of authenticated network packets [19], to provide a holistic approach for security-aware scheduling in distributed embedded control.

REFERENCES

[1] A. Anta and P. Tabuada. 2009. On the Benefits of Relaxing the Periodicity Assumption for Networked Control Systems over CAN. In *2009 30th IEEE Real-Time Systems Symposium*. 3–12. https://doi.org/10.1109/RTSS.2009.39

[2] A. Anta and P. Tabuada. 2010. To Sample or not to Sample: Self-Triggered Control for Nonlinear Systems. *IEEE Trans. Automat. Control* 55, 9 (Sept 2010), 2030–2042. https://doi.org/10.1109/TAC.2010.2042980

[3] Sanjoy Baruah and Enrico Bini. 2008. Partitioned scheduling of sporadic task systems: an ILP-based approach. *Proceedings of the International Conference on Design and Architectures for Signal and Image Processing (DASIP)* (2008).

[4] Sanjoy Baruah, Deji Chen, Sergey Gorinsky, and Aloysius Mok. 1999. Generalized Multiframe Tasks. *Real-Time Systems* 17, 1 (01 Jul 1999), 5–22. https://doi.org/10.1023/A:1008030427220

[5] Sanjoy K. Baruah, Louis E. Rosier, and Rodney R. Howell. 1990. Algorithms and complexity concerning the preemptive scheduling of periodic, real-time tasks on one processor. *Real-Time Systems* 2, 4 (01 Nov 1990), 301–324. https://doi.org/10.1007/BF01995675

[6] Pietro Belotti, Pierre Bonami, Matteo Fischetti, Andrea Lodi, Michele Monaci, Amaya Nogales-Gómez, and Domenico Salvagnin. 2016. On handling indicator constraints in mixed integer programming. *Computational Optimization and Applications* 65, 3 (01 Dec 2016), 545–566. https://doi.org/10.1007/s10589-016-9847-8

[7] Enrico Bini and Giorgio Buttazzo. 2009. The space of EDF deadlines: the exact region and a convex approximation. *Real-Time Systems* 41, 1 (01 Jan 2009), 27–51. https://doi.org/10.1007/s11241-008-9060-7

[8] G. C. Buttazzo. 2011. *Hard Real-Time Computing Systems: Predictable Scheduling Algorithms and Applications* (3rd ed.). Springer, 110–114. https://doi.org/10.1007/978-1-4614-0676-1

[9] Giorgio C. Buttazzo, Giuseppe Lipari, Marco Caccamo, and Luca Abeni. 2002. Elastic Scheduling for Flexible Workload Management. *IEEE Trans. Comput.* 51, 3 (March 2002), 289–302. https://doi.org/10.1109/12.990127

[10] Anton Cervin, Johan Eker, Bo Bernhardsson, and Karl-Erik Årzén. 2002. Feedback–Feedforward Scheduling of Control Tasks. *Real-Time Systems* 23, 1 (01 Jul 2002), 25–53. https://doi.org/10.1023/A:1015394302429

[11] Stephen Checkoway, Damon McCoy, Brian Kantor, Danny Anderson, Hovav Shacham, Stefan Savage, Karl Koscher, Alexei Czeskis, Franziska Roesner, and Tadayoshi Kohno. 2011. Comprehensive Experimental Analyses of Automotive Attack Surfaces. In *Proceedings of the 20th USENIX Conference on Security (SEC'11)*. USENIX Association, Berkeley, CA, USA, 6–6. http://dl.acm.org/citation.cfm?id=2028067.2028073

[12] T. M. Chen and S. Abu-Nimeh. 2011. Lessons from Stuxnet. *Computer* 44, 4 (April 2011), 91–93. https://doi.org/10.1109/MC.2011.115

[13] A. Greenberg. 2015. Hackers Remotely Kill a Jeep on the Highway, Wired Magazine. (2015).

[14] M. Hasan, S. Mohan, R. B. Bobba, and R. Pellizzoni. 2016. Exploring Opportunistic Execution for Integrating Security into Legacy Hard Real-Time Systems. In *2016 IEEE Real-Time Systems Symposium (RTSS)*. 123–134. https://doi.org/10.1109/RTSS.2016.021

[15] I. Jovanov and M. Pajic. 2017. Relaxing Integrity Requirements for Resilient Control Systems. *CoRR* abs/1707.02950 (2017). https://arxiv.org/abs/1707.02950

[16] I. Jovanov and M. Pajic. 2017. Sporadic Data Integrity for Secure State Estimation. In *55th IEEE Conference on Decision and Control (CDC)*.

[17] Andrew J. Kerns, Daniel P. Shepard, Jahshan A. Bhatti, and Todd E. Humphreys. 2014. Unmanned Aircraft Capture and Control Via GPS Spoofing. *J. Field Robot.* 31, 4 (July 2014), 617–636. https://doi.org/10.1002/rob.21513

[18] C. Kwon, W. Liu, and I. Hwang. 2014. Analysis and Design of Stealthy Cyber Attacks on Unmanned Aerial Systems. *Journal of Aerospace Information Systems* 11, 8 (2014), 525–539. https://doi.org/10.2514/1.I010201

[19] V. Lesi, I. Jovanov, and M. Pajic. 2017. Network Scheduling for Secure Cyber-Physical Systems. In *38th IEEE Real-Time Systems Symposium (RTSS)*.

[20] Joseph Y.-T. Leung and M.L. Merrill. 1980. A note on preemptive scheduling of periodic, real-time tasks. *Inform. Process. Lett.* 11, 3 (1980), 115 – 118. https://doi.org/10.1016/0020-0190(80)90123-4

[21] Chung-Wei Lin, Bowen Zheng, Qi Zhu, and Alberto Sangiovanni-Vincentelli. 2015. Security-Aware Design Methodology and Optimization for Automotive Systems. *ACM Trans. Des. Autom. Electron. Syst.* 21, 1, Article 18 (Dec. 2015), 26 pages. https://doi.org/10.1145/2803174

[22] M. Lin, L. Xu, L. T. Yang, X. Qin, N. Zheng, Z. Wu, and M. Qiu. 2009. Static Security Optimization for Real-Time Systems. *IEEE Transactions on Industrial Informatics* 5, 1 (Feb 2009), 22–37. https://doi.org/10.1109/TII.2009.2014055

[23] Yuriy Zacchia Lun, Alessandro D'Innocenzo, Ivano Malavolta, and Maria Domenica Di Benedetto. 2016. Cyber-Physical Systems Security: a Systematic Mapping Study. *CoRR* abs/1605.09641 (2016). http://arxiv.org/abs/1605.09641

[24] Y. Mo, E. Garone, A. Casavola, and B. Sinopoli. 2010. False data injection attacks against state estimation in wireless sensor networks. In *49th IEEE Conference on Decision and Control (CDC)*. 5967–5972. https://doi.org/10.1109/CDC.2010.5718158

[25] A. K. Mok and D. Chen. 1996. A multiframe model for real-time tasks. (Dec 1996), 22–29. https://doi.org/10.1109/REAL.1996.563696

[26] Gurobi Optimization Inc. 2014. Gurobi optimizer reference manual. (2014). http://www.gurobi.com

[27] M. Pajic, I. Lee, and G. J. Pappas. 2017. Attack-Resilient State Estimation for Noisy Dynamical Systems. *IEEE Transactions on Control of Network Systems* 4, 1 (March 2017), 82–92. https://doi.org/10.1109/TCNS.2016.2607420

[28] M. Pajic, J. Weimer, N. Bezzo, P. Tabuada, O. Sokolsky, I. Lee, and G. J. Pappas. 2014. Robustness of attack-resilient state estimators. In *2014 ACM/IEEE International Conference on Cyber-Physical Systems (ICCPS)*. 163–174. https://doi.org/10.1109/ICCPS.2014.6843720

[29] B. Peng and N. Fisher. 2016. Parameter Adaption for Generalized Multiframe Tasks and Applications to Self-Suspending Tasks. In *2016 IEEE 22nd International Conference on Embedded and Real-Time Computing Systems and Applications (RTCSA)*. 49–58. https://doi.org/10.1109/RTCSA.2016.15

[30] M. Pettersson. 1997. *Driveline modeling and control.* Ph.D. Dissertation. Department of Electrical Engineering, Linköping University.
[31] C. T. Wei. 2009. *Modeling and control of an engine fuel injection system.* Master's thesis.
[32] Tao Xie and Xiao Qin. 2007. Improving Security for Periodic Tasks in Embedded Systems Through Scheduling. *ACM Trans. Embed. Comput. Syst.* 6, 3, Article 20 (July 2007). https://doi.org/10.1145/1275986.1275992
[33] B. Zheng, P. Deng, R. Anguluri, Q. Zhu, and F. Pasqualetti. 2016. Cross-Layer Codesign for Secure Cyber-Physical Systems. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 35, 5 (May 2016), 699–711. https://doi.org/10.1109/TCAD.2016.2523937

Received April 2017; revised May 2017; accepted June 2017