Security-driven Codesign with Weakly-hard Constraints for Real-time Embedded Systems

Hengyi Liang*, Zhilu Wang*, Debayan Roy†, Soumyajit Dey‡, Samarjit Chakraborty†, Qi Zhu*
* Northwestern University, USA. Emails: {hengyiliang2018@u.,zhiluwang2018@u.,qzhu@}northwestern.edu
† Technical University of Munich, Germany. Emails: {debayan.roy,samarjit}@tum.de
‡ Indian Institute of Technology, India. Email: soumya@cse.iitkgp.ac.in

***Abstract*—For many embedded systems, such as automotive electronic systems, security has become a pressing challenge. Limited resources and tight timing constraints often make it difficult to apply even lightweight authentication and intrusion detection schemes, especially when retrofitting existing designs. Moreover, traditional hard deadline assumption is insufficient to describe control tasks that have certain degrees of robustness and can tolerate some deadline misses while satisfying functional properties such as stability. In this work, we explore feasible weakly-hard constraints on control tasks, and then leverage the scheduling flexibility from those allowed misses to enhance system's capability for accommodating security monitoring tasks. We develop a co-design approach that 1) sets feasible weakly-hard constraints on control tasks based on quantitative analysis, ensuring the satisfaction of control stability and performance requirements; and 2) optimizes the allocation, priority, and period assignment of security monitoring tasks, improving system security while meeting timing constraints (including the weakly-hard constraints on control tasks). Experimental results on an industrial case study and a set of synthetic examples demonstrated the significant potential of leveraging weakly-hard constraints to improve security and the effectiveness of our approach in exploring the design space to fully realize such potential.**

I. Introduction

Security challenges in automotive systems: Security has become a pressing issue for automotive electronic systems. Researchers have demonstrated that malicious attackers can successfully compromise a variety of local and remote interfaces in vehicles, and then carry out attacks on safety-critical components via in-vehicle communication networks such as the Controller Area Networks (CAN) buses [1], [2], [3].

As one important element to address automotive security challenges, various approaches have been proposed to harden in-vehicle communications, via authentication mechanisms [4], [5], [6] or intrusion detection techniques [7], [8], [9], [10]. For instance, in [8], an anomaly-based intrusion detection technique is proposed to analyze the local clocks of Electronic Control Units (ECUs) for identifying abnormal behavior. In [7], a detection technique is proposed to constantly monitor in-vehicle communication (e.g., CAN messages) and detect malicious attacks that alter the message streams.

However, due to limited resources and tight timing constraints, it is often challenging or even prohibitive to apply these intrusion detection or authentication techniques. The addition of these security mechanisms may prolong the execution of other functions (e.g., by preempting the existing control tasks), and cause them to violate their execution deadlines. Studies in [6], [11], [12], [8] showed that it is difficult to apply even lightweight authentication methods in current in-vehicle networks such as CAN, and adding intrusion detection techniques will also require careful timing analysis and may not be feasible under hard timing constraints [7], [13].

Control with deadline misses and weakly-hard constraints: Many system functions such as some control tasks, however, have certain degrees of robustness and can tolerate some deadline misses while still satisfy functional properties and performance requirements, as long as those misses are *bounded and dependably controlled*. Recent works have studied the impact of deadline misses on control performance and stability [14], [15], [16], [17], [18]. For instance, in [15], the authors present an analytical bound of deadline miss ratio that can ensure the stability of a distributed embedded controller. In [16], the worst-case control performance for an LQR controller is analyzed under deadline misses. In [14], the authors provide a general framework to capture the control performance with respect to a specific sequence of deadline miss pattern.

Weakly-hard constraints are thus proposed to capture the timing requirements for tasks that allow deadline misses. A common form is the (k, N) constraint, which specifies that among any N consecutive task executions, at most k instances can miss their deadlines [19], [20]. These constraints more precisely reflect the timing requirements for many system functions, and provide more scheduling slack by allowing deadline misses with safety guarantees [21], [22]. In the literature, methods were developed for leveraging the additional slack to improve schedulability [19], [23], [24], [25]. In [19], weakly-hard constraints are formally defined and algorithms are proposed for scheduling periodic tasks under such constraints. In [25], schedulability analysis for periodic tasks is improved with unknown task activation offsets. In [23], a general non-periodic task model is defined and a typical worst-case analysis (TWCA) algorithm is introduced for analyzing sporadic overloads. TWCA for weakly-hard schedulability analysis on the general model is formally presented in [24], and extended for systems with task dependencies in [26]. There are also related works that leverage weakly-hard constraints for better energy-driven scheduling, such as [27], [28].

Our codesign approach in leveraging weakly-hard constraints to enhance security: In this work, we leverage the scheduling flexibility/slack from the control tasks that

allow deadline misses to improve the security of automotive systems, i.e., to better accommodate security monitoring tasks while meeting the stability and performance requirements of control tasks. We develop a codesign approach that holistically addresses control and security, bridging the two aspects via the exploration of weakly-hard constraints on control tasks and the design of security monitoring tasks (the general codesign methodology has been applied to hard real-time systems in our prior work [29], [30], [31]). More specifically, in this work we model the control tasks running on ECUs as switched systems and derive the control stability and performance under various deadline miss patterns within a hyper-period. We then consider deploying security monitoring tasks on the ECUs and let them monitor CAN messages for anomaly detection. We developed an algorithm to explore the allocation, priority, and period assignment of these security monitoring tasks, with control analysis and schedulabiltiy analysis, to improve system security while ensuring that 1) for control tasks that allow deadline misses, their stability and performance requirements are met based on the analysis of the deadline miss patterns, 2) for other tasks including the security monitoring tasks, their deadlines are always met, and 3) constraints on security monitoring are met.

Our approach could free up resources for incorporating security monitoring mechanisms into existing automotive system designs. This is critical in automotive domain, as there are often significant reuse of legacy E/E architectures and constant needs of updating existing systems or system designs for fixing bugs and providing new functionality (e.g., security features) [32], [33], [34]. While this work focuses on automotive systems, our approach could be modified to address other resource-constrained systems that have similar characteristics and allow weakly-hard constraints. To the best of our knowledge, this work is the first to leverage weakly-hard constraints for control tasks to harden system security. The main technical contribution of this work includes:

- We presented a control analysis method for linear time-invariant control systems, to verify the stability and measure the control performance under different deadline miss patterns. We also formulated the addition of security monitoring tasks for intrusion detection, and defined a corresponding security objective function and other related constraints.
- We formulated a constrained multi-objective optimization problem to determine the configuration of security monitoring tasks for improving system security and control performance. The two objectives are typically conflicting and the problem is non-linear and non-convex. Thus, we developed a meta-heuristic approach (i.e., simulated annealing) for exploring the design space and obtaining the pseudo Pareto-optimal configurations for security monitoring tasks.
- We conducted experiments on an industrial case study and a set of synthetic examples. The experiments demonstrated that 1) compared with hard deadlines, weakly-hard constraints can significantly improve system's ability to accommodate security monitoring tasks, and 2) our codesign approach is effective for exploring the design space and fully leveraging the potential of weakly-hard constraints.

In the following, Section II describes our system model. Section III introduces our problem analysis and formulation. Section IV presents our algorithm for exploring the design of security monitoring tasks, with control and schedulability analysis. Section V presents the experimental results.

II. System Modeling

We consider a system that consists of multiple homogeneous single-core ECUs connected with a CAN bus, and let $\mathcal{E} = \{e_1, e_2, \ldots, e_{n_e}\}$ denote the set of ECUs. As shown in Fig. 1, different types of tasks may run on the ECUs (similarly as assumed in [7]), including: **a)** security monitoring tasks that monitor the CAN messages for intrusion detection, **b)** control tasks that implement controller functions, and **c)** tasks that implement other functionalities.

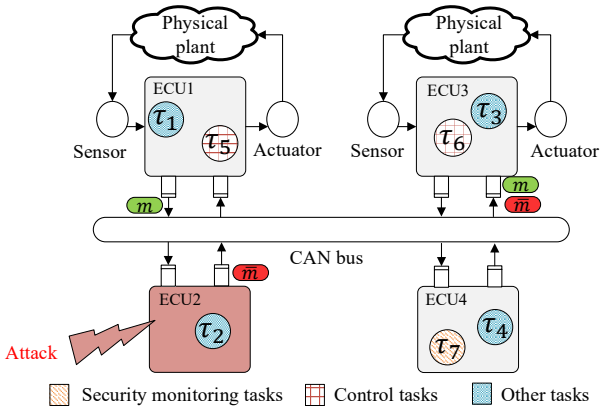

Fig. 1. System model.

We let $\mathcal{T} = \{\tau_1, \tau_2, \ldots, \tau_n\}$ denote the set of all the tasks in the system. We consider that all tasks are periodic, and tasks running on the same ECU are scheduled based on preemptive static-priority based policy. Each task τ_i can be represented by a tuple $\{c_{\tau_i}, d_{\tau_i}, t_{\tau_i}, p_{\tau_i}\}$, where c_{τ_i} is the worst-case execution time (WCET), d_{τ_i} is the deadline, t_{τ_i} is the task period, and p_{τ_i} is the task priority. We assume that some of the control tasks may allow deadline misses in a bounded and controlled manner, and their timing requirements can be captured by weakly-hard constraints as introduced later. For the rest of the tasks, including security monitoring tasks, other tasks, and control tasks that do not allow deadline misses, we assume hard timing constraints, i.e., their deadlines should always be met[1]. Let $\mathcal{M} = \{m_1, m_2, \ldots, m_{n_{msg}}\}$ denote the set of CAN bus messages. Each message m_i has a period t_{m_i}. In this work, we assume that the configuration of CAN bus messages is given by the system designers and is schedulable. An example of the system model is explained in Appendix A.

[1]In principle, other tasks (e.g., those for sensing, computation and communication), security monitoring tasks, and CAN messages may also allow deadline misses and be defined with weakly-hard constraints. Those scenarios are beyond the scope of this paper, and will be addressed in future work.

Next, we introduce our models for the security monitoring tasks and control tasks in more details.

A. Security Monitoring Tasks

As stated above, we consider security monitoring tasks that monitor CAN messages and implement certain anomaly-based intrusion detection mechanism, such as those in [10], [35], [36], [7], [8]. Our model is relatively general, and does not restrict the monitoring tasks to a specific intrusion detection mechanism. We assume that on an ECU with security monitoring tasks, a local buffer will store the messages being monitored and other relevant information, e.g., the timestamps of the message transmissions. At each activation of a security monitoring task, it retrieves the messages it monitors and other corresponding information, and checks any possible anomalies. Note that as messages are periodically transmitted, the security monitoring tasks in fact monitor message streams.

An ECU may have multiple security monitoring tasks that scrutinize different CAN messages. We let $TS_i = \{\tau_{monitor,i,1}, \ldots, \tau_{monitor,i,|TS_i|}\}$ denote the set of monitoring tasks on ECU e_i, where operator $|\bullet|$ denotes the cardinality of a set. The set of all the security monitoring tasks in the system is denoted as $\mathcal{T}_S = \bigcup_{i=1}^{n_e} TS_i$. Each security monitoring task $\tau_{monitor,i,j}$ is captured with a period $t_{\tau_{monitor,i,j}}$, a deadline equal to its period, a worst-case execution time $c_{\tau_{monitor,i,j}}$, and a priority $p_{\tau_{monitor,i,j}}$. As stated before, the security monitoring tasks are scheduled together with control tasks and other tasks based on static priorities.

The task $\tau_{monitor,i,j}$ can monitor a set of messages that have the same period, denoted by $M_{\tau_{monitor,i,j}}$. We use $tm_{i,j}$ to denote the period of these messages. Since a security monitoring task would typically inspect each monitored message and perform some computation based on it, it is conceivable that the WCET of such a task $\tau_{monitor,i,j}$ will be polynomial in $|M_{\tau_{monitor,i,j}}|$. In this work, we simply consider

$$c_{\tau_{monitor,i,j}} = k_{wcet} \times |M_{\tau_{monitor,i,j}}|, \tag{1}$$

where k_{wcet} is considered as a given parameter that depends on the monitoring functionality. In our experiments, we relate the computation time of security monitoring tasks to a prior study in the literature [37], [7], where it shows monitoring 15 message streams will consume about 5% of the ECU resource. Therefore, we assume that $\frac{1 \times k_{wcet}}{t_{\tau_{monitor,i,j}}}$ will take $\frac{5\%}{15} = 0.33\%$ of ECU utilization in experiments.

B. Controller Model and Control Tasks

In this work, we study linear time-invariant (LTI) systems for which the dynamics of the physical plant can be modeled as $\dot{x}(t) = Ax(t) + Bu(t)$, $y(t) = Cx(t)$. Here, A, B and C are system matrices, and $x(t)$, $u(t)$ and $y(t)$ are vectors representing the system state, control input and system output at time t, respectively. We consider that the system state is read periodically by sensing devices at discrete time instants $\{t_k\}$, where the sampling period is a constant and is given by $h = t_{k+1} - t_k$. The system state $x[k]$ (i.e., $x(t_k)$) read at time t_k is used to compute the control input $u[k]$ which is then applied to the plant by the actuator. As the controller is implemented on an embedded platform, computation and communication can take non-negligible time which may result in a sensing-to-actuation delay. We assume that the sensing-to-actuation delay is bounded by a deadline D and $D < h$ (we will discuss later how this deadline may be evaluated). We further assume that the system adopts the Logical Execution Time (LET) paradigm [38], where the control input is updated at the sensing-to-actuation deadlines. The LET implementation provides fixed closed-loop delay (when deadline is met) and thus facilitates more predictable control, and has been adopted in the literature [18], [14]. In our case, the actuator applies the control input $\{u[k]\}$ at time instants $\{t_k + D\}$.

Corresponding to the sampling period h and the sensing-to-actuation delay D, we can write the equivalent delayed discrete-time system model (Chapter 2 in [39]) as

$$x[k+1] = A_d x[k] + B_{d,0} u[k] + B_{d,1} u[k-1], \; y[k] = Cx[k], \tag{2}$$

where $B_{d,0} = \int_0^{h-D} e^{As} \cdot B ds$ and $B_{d,1} = \int_{h-D}^{h} e^{As} \cdot B ds$.

For an augmented state vector $z[k] = \begin{bmatrix} x[k] \\ u[k-1] \end{bmatrix}$, we reformulate (2) as

$$z[k+1] = A_{aug} z[k] + B_{aug} u[k], \; y[k] = C_{aug} z[k], \tag{3}$$

where $A_{aug} = \begin{bmatrix} A_d & B_{d,1} \\ \mathbf{0} & \mathbf{0} \end{bmatrix}$, $B_{aug} = \begin{bmatrix} B_{d,0} \\ \mathbf{I} \end{bmatrix}$, $C_{aug} = \begin{bmatrix} C & \mathbf{0} \end{bmatrix}$ with $\mathbf{0}$ and $\mathbf{I}$ denoting zero matrix and identity matrix respectively of suitable dimensions [40]. We consider the feedback control law to be $u[k] = -Kz[k]$ with the feedback gain K calculated using pole-placement technique [41].

Controller implementation: We consider that the control law is implemented using a software control task τ_c running on an ECU. As shown in Fig. 1, each controlled plant is connected locally to the sensing and actuation units, and we assume that the sensing and actuation delays are bounded by d_s and d_a, respectively[2]. Thus, to realize the sensing-to-actuation delay D, the deadline of the control task d_{τ_c} is restricted by $d_{\tau_c} = D - d_s - d_a$. Now, during execution, if a job of the control task τ_c meets the deadline d_{τ_c}, the control input will be applied with the delay D (following the LET paradigm as discussed before); if a job of τ_c violates the deadline d_{τ_c}, the job will be killed and the control input will be zero.

A similar control strategy is used in [15], with the assumption that the sensing-to-actuation delay is exactly equal to one sampling period. In [14], the control input is held when a job is killed, assuming that the maximum consecutive deadline misses is upper-bounded and the magnitude of a disturbance is known. In our work, these assumptions are relaxed. Next in Section III-A, we introduce a control performance metric to measure the ability of the system to quickly reject a disturbance, and thereafter, show how to evaluate the impact of deadline misses on stability and control performance.

[2]In the cases where sensing and actuation units are distributed and communicated to control tasks via CAN messages, timing analysis on the messages should be conducted to bound the worst-case sensing and actuation delays.

III. Problem Analysis and Formulation

In our work, we consider adding security monitoring tasks to an existing system design, and thus assume the allocation, priority, and period of control tasks and other tasks are given. We focus on exploring the allocation, priority, and period of security monitoring tasks, to improve system security while ensuring that various constraints are met. To achieve this, we developed a holistic formulation to model and analyze the control stability and performance under deadline misses, the constraints related to security monitoring tasks, a security objective function for measuring system's intrusion detection capability, and the schedulability constraints for all tasks (under weakly-hard or hard deadlines), as introduced below.

A. Stability and Control Performance

Asymptotic stability: A discrete-time system is asymptotically stable if all the closed-loop poles lie inside a unit circle (Chapter 3 in [39]). From the augmented state-space model (3) and the control law, we have the following closed loop dynamics:

$$z[k+1] = (A_{aug} - B_{aug}K)z[k] = A_{cl}z[k]. \tag{4}$$

The system, described in (4), is asymptotically stable, if and only if all eigenvalues of A_{cl} satisfy $|\lambda_i| < 1$ [39].

Control performance metric: In this work, we consider stabilization control, i.e., the controller must *quickly* bring the system back to the equilibrium state after a disturbance. Without loss of generality, we assume that the equilibrium state is at the origin of the state-space. Let us consider that a disturbance arriving at time t_k brings the system to a state $z[k]$. The magnitude of the disturbance is measured by the deviation of the system from the equilibrium state, i.e., $\|z[k]\|$. The system state at time t_{k+r} is $z[k+r]$. Assuming no further disturbance injection, the residual disturbance J_r after r sampling intervals is given by $J_r = \frac{\|z[k+r]\|}{\|z[k]\|}$. We quantify the ability of a controller to reject disturbances using a metric H where H is the number of sampling intervals needed to bring the residual disturbance J_r to less than or equal to a certain threshold J_{th}, i.e.,

$$J_r \leq J_{th}, \quad \forall r \geq H. \tag{5}$$

The lower the value of H is, the quicker the disturbance is rejected and the better the control performance is. With no overlapping disturbances, we can write $z[k+r] = A_{cl}^r z[k]$, and therefore, $J_r \leq \|A_{cl}^r\|$. As J_r depends on $z[k]$, which is a variable and not known in advance, we will use this upper bound to evaluate the performance of a controller. We consider that the controller must satisfy a certain minimum performance requirement, denoted as H^r, such that $H \leq H^r$.

Deadline hit/miss pattern: For the problem under study, given the set of control and monitoring tasks for the target architecture and the task configuration, we can obtain a pattern of deadline hits/misses for each task using schedulability analysis (as introduced later in Section IV-A). The periodicity of the deadline hit/miss sequence is determined by the hyper-period of all the tasks mapped on the same ECU, i.e., the least common multiple of these tasks' periods. Here, a binary variable σ_n is used to denote the state of the n-th job of a task in a hyper-period, i.e., $\sigma_n = 1$ when the deadline is met (hit) while $\sigma_n = 0$ when the deadline is missed. Let $\Pi = (\sigma_1\sigma_2\cdots\sigma_N\cdots)$ denote a periodic deadline hit/miss pattern for a control task, where, there are N sampling instants within a hyper-period. For such an infinitely repeating pattern of deadline hits/misses, there exist a maximum of N distinct control sequences for a given disturbance. For an example pattern $\Pi = (10111011\cdots)$, disturbance arriving at the first, second, third and fourth sampling instants will result in four different control sequences. This is because the sequence of deadline hits/misses starting from the instant where the disturbance is injected is different in each of the cases, i.e., $\Pi_1 = (10111011\cdots)$, $\Pi_2 = (01110111\cdots)$, $\Pi_3 = (11101110\cdots)$ and $\Pi_4 = (11011101\cdots)$. We denote such a sequence starting from the n-th sampling instant in a hyper-period as $\Pi_n = (\sigma_n\sigma_{n+1}\cdots\sigma_N\sigma_1\cdots\sigma_{n-1}\sigma_n\cdots)$.

Switched system model: According to the control model discussed in Section II-B, $u[k] = 0$ when $\sigma_{mod(k,N)+1} = 0$ and $u[k] = -Kz[k]$ when $\sigma_{mod(k,N)+1} = 1$, where $mod(k,N)$ gives the remainder when k is divided by N. The closed-loop system can then be modeled as a switched system depending on the deadline hit/miss pattern. Based on (2), (3) and the adopted control model, the switched system is composed of a maximum of four subsystems. Here we do not assume any restriction on the number of consecutive deadline misses (contrary to [14]). These subsystems can be modeled as:

- When $\sigma_{mod(k,N)+1} = 0$ and $\sigma_{mod(k-1,N)+1} = 0$, (2) becomes $x[k+1] = A_d x[k]$ and $u[k] = 0$. Correspondingly, the augmented state-space model in (3) becomes
$$z[k+1] = \begin{bmatrix} A_d & \mathbf{0} \\ \mathbf{0} & \mathbf{0} \end{bmatrix} z[k] = A_{00}z[k].$$
- With $\sigma_{mod(k,N)+1} = 1$ and $\sigma_{mod(k-1,N)+1} = 0$, (2) becomes $x[k+1] = A_d x[k] + B_{d,0}u[k]$ and $u[k] = -Kz[k]$. Thus, the augmented state-space model in (3) becomes
$$z[k+1] = \left(\begin{bmatrix} A_d & \mathbf{0} \\ \mathbf{0} & \mathbf{0} \end{bmatrix} - \begin{bmatrix} B_{d,0} \\ 1 \end{bmatrix} K \right) z[k] = A_{01}z[k].$$
- For $\sigma_{mod(k,N)+1} = 0$ and $\sigma_{mod(k-1,N)+1} = 1$, (2) becomes $x[k+1] = A_d x[k] + B_{d,1}u[k-1]$ and $u[k] = 0$. The augmented state-space model in (3), therefore, becomes
$$z[k+1] = \begin{bmatrix} A_d & B_{d,1} \\ \mathbf{0} & \mathbf{0} \end{bmatrix} z[k] = A_{10}z[k].$$
- In case, $\sigma_{mod(k,N)+1} = 1$ and $\sigma_{mod(k-1,N)+1} = 1$, the plant dynamics evolve as in (2). The augmented state-space model is, thus, as given in (3). The state-transition matrix A_{11} can be written as $A_{11} = A_{aug} - B_{aug}K$.

Stability and performance constraints: For a deadline hit/miss pattern Π, we need to evaluate the stability and performance of the control loop for all possible sequences of hits/misses, i.e., $\{\Pi_1, \Pi_2, \cdots, \Pi_N\}$. Given a sequence Π_n of deadline hits/misses, the evolution of the system from $z[k]$ (where k is the n-th sampling instant in a hyper-

period consisting of N samples, i.e., $n = mod(k, N) + 1$) would be governed by the sequence of closed-loop matrices $(A_{\sigma_{n-1}\sigma_n}, A_{\sigma_n\sigma_{n+1}}, \cdots, A_{\sigma_N\sigma_1}, A_{\sigma_1\sigma_2}, \cdots, A_{\sigma_{n-2}\sigma_{n-1}}, \cdots)$. Therefore, the system evolves as follows:

$$\begin{aligned} &z[k+r] = A_{n,r}z[k], \quad \text{where,} \\ &A_{n,1} = A_{\sigma_{mod(k-1,N)+1}\sigma_{mod(k,N)+1}}, \\ &A_{n,r} = A_{\sigma_{mod(k+r-2,N)+1}\sigma_{mod(k+r-1,N)+1}} \cdot A_{n,r-1}. \end{aligned} \tag{6}$$

Such a switched system is asymptotically stable if the eigenvalues of $A_{n,N}$ lie inside a unit circle. Thus, the stability constraint for a control loop experiencing a deadline hit/miss according to Π is given by

$$\forall 1 \leq n \leq N, \quad \forall \lambda_i \in eig(A_{n,N}), \quad |\lambda_i| < 1, \tag{7}$$

where $eig(A_{n,N})$ is the set of eigenvalues of $A_{n,N}$.

When the disturbance is observed at the k-th instant and $n = mod(k, N) + 1$, the residual disturbance $J_r(n)$ after r time samples can be written as $J_r(n) = \frac{\|z[k+r]\|}{\|z[k]\|} \leq \|A_{n,r}\|$. Now the control performance as measured by $H(n)$ for the sequence Π_n is $J_r(n) \leq \|A_{n,r}\| \leq J_{th}, \forall r \geq H(n)$ and the worst-case performance H^* for a given periodic sequence Π must satisfy a given requirement H^r and is given by

$$H^* = \max_{1 \leq n \leq N} H(n) \leq H^r. \tag{8}$$

If (8) is not satisfied then Π is not an acceptable sequence.

Control performance objective: During the design space exploration, we consider control performance as an optimization objective, and address it together with security. We let $H^{des}_{\tau_i}$ denote the control performance when no deadline miss occurs (for normalization purpose), and let $H^*_{\tau_i}$ be computed as in (8). We define the system-level control performance P as follows:

$$P = \sum_{\tau_i \in \mathcal{T}_C} \eta_i \frac{H^*_{\tau_i}}{H^{des}_{\tau_i}}, \tag{9}$$

where η_i are the weights and $\mathcal{T}_C$ is the set of control tasks.

B. Security Constraints and Objective

Security monitoring constraints: We consider the following requirements when adding security monitoring tasks:

- *Coverage:* A pre-defined set of critical CAN messages, denoted by $\mathcal{M}_{cri}$, may be given. Each message in $\mathcal{M}_{cri}$ should be covered by at least one security monitoring task.
- *Redundancy:* A CAN message may be required to be monitored by multiple tasks on *different* ECUs to avoid a single point of failure.

We can formalize the above requirements for message m_i as

$$\sum_{j=1}^{n_e} a_{i,j} = \mu_i, \quad \forall m_i \in \mathcal{M}_{cri}, \tag{10}$$

where binary variable $a_{i,j} = 1$ if m_i is monitored by a security task on ECU e_j. Parameter μ_i is defined as the redundancy level for m_i ($\mu_i \geq 1$), representing how many tasks on different ECUs are required to monitor m_i.

Moreover, we may set constraints on the period of security monitoring tasks. Intuitively, setting a smaller period (higher frequency) for a security monitoring task $\tau_{monitor,i,j}$ may provide better intrusion detection capability (as further discussed below in security objective), but also incur higher computational overhead. To balance the detection efficacy and overhead, we set a period constraint as

$$T^{des}_{i,j} \leq t_{monitor,i,j} \leq T^{max}_{i,j}, \tag{11}$$

where $T^{des}_{i,j}$ is the *desired monitoring period* that provides ideal detection efficacy, while $T^{max}_{i,j}$ is the maximum period that can still provide meaningful detection capability. Similar constraints were used in the literature [13], [42], with $T^{des}_{i,j}$ and $T^{max}_{i,j}$ assumed as given by the designers. In our experiments, $T^{des}_{i,j}$ is set to $tm_{i,j}$, the period of monitored messages. $T^{max}_{i,j}$ is set as an integer multiple of $tm_{i,j}$, i.e., $T^{max}_{i,j} = K_{max} tm_{i,j}$.

Security objective: To define a system-level security objective function, we first measure the efficacy of each security monitoring task $\tau_{monitor,i,j}$ to detect the potential anomaly of a message $m_k \in M_{\tau_{monitor,i,j}}$. We assume the anomaly detection mechanism is already given and only focus on its timing aspect, and we define a worst-case detection delay metric $D_{\tau_{monitor,i,j}}$ to measure the maximum time it takes for the monitoring task to detect the message anomaly.

Consider a security attack occurs and leads to abnormal behavior for m_k at time t. In the worst case, this could happen right after the s-th job instance of the monitoring task $\tau_{monitor,i,j}$ had just been activated, and we have to wait until the $(s+1)$-th job of the monitoring task to be activated *and* complete its execution for detection. Thus, the worst-case detection delay $D_{\tau_{monitor,i,j}}$ should include the monitoring task's activation period and worst-case response time[3] , i.e., $D_{\tau_{monitor,i,j}} = t_{\tau_{monitor,i,j}} + r_{\tau_{monitor,i,j}}$.

Since a message m_k may be monitored by multiple security monitoring tasks, we can further define the worst-case detection delay for message m_k as the maximum time for the anomaly of m_k to be detected by any one of its monitoring tasks, i.e., $D_{m_k} = \min\{D_{\tau_{monitor,i,j}} | m_k \in M_{\tau_{monitor,i,j}}\}$. For normalization, we define the desired detection delay as $D^{des}_{m_k} = \min\{D^{des}_{\tau_{monitor,i,j}} | m_k \in M_{\tau_{monitor,i,j}}\}$, where $D^{des}_{\tau_{monitor,i,j}} = T^{des}_{i,j} + c_{\tau_{monitor,i,j}}$. That is, $D^{des}_{\tau_{monitor,i,j}}$ corresponds to the scenario where task $\tau_{monitor,i,j}$ is assigned with the desired period and highest priority. With this, we define a system-level security objective as

$$\mathcal{S} = \sum_{m_k \in \mathcal{M}_{cri}} \omega_k \frac{D_{m_k}}{D^{des}_{m_k}}, \tag{12}$$

where ω_k are the weighting factors.

C. Schedulability Constraints

As stated in Section II, some control tasks in the system may be bounded by weakly-hard constraints, while the rest of the tasks are bounded by hard deadlines. For a control task τ_i with weakly-hard constraints, we denote its timing requirement

[3]Here we assume the security monitoring task is able to detect the anomaly in its first activation. This may not be the case in complex scenarios, but as our work is agnostic to the specific anomaly detection functionality, we believe this is a reasonable first-degree approximation.

as $\zeta_i = \{(k_i^1, N_i^1), \ldots, (k_i^{n_i}, N_i^{n_i})\}$ (similar to [14]), where (k_i^j, N_i^j) means for any N_i^j consecutive activations of task τ_i, at most k_i^j deadline misses are allowed. Task τ_i satisfies its weakly-hard constraint, i.e., is schedulable, if

$$dmm_i(N_i^j) \leq k_i^j, \quad \forall j, 1 \leq j \leq n_i, \tag{13}$$

where $dmm_i(N_i^j)$ denotes the worst-case number of deadline misses of τ_i in N_i^j consecutive activations. In Section IV, we present our event-based simulation method for analyzing the schedule within a hyper-period and checking the deadline misses (i.e., $dmm_i(N_i^j)$) for tasks.

For a task τ_i with hard timing constrains, we can calculate its worst-case response time r_{τ_i} using the busy window analysis in [23]. The task is schedulable if

$$r_{\tau_i} \leq d_{\tau_i}. \tag{14}$$

D. Overall Formulation

Our codesign approach optimizes a joint objective function that combines the security objective in (12) and the control performance objective in (9), as

$$\mathcal{J} = \alpha S + \beta P, \tag{15}$$

where α and β are parameters for trading off the two objectives. The overall constrained optimization formulation with constraints on control stability, performance, security monitoring, and schedulability is:

minimize $\mathcal{J}$

subject to Equations (7), (8), (10), (11), (13), (14).

IV. Optimization Algorithm for Codesign

The optimization problem formulated above in Section III-D is non-convex and complex to solve. In particular, the stability and performance for each control task depend on its deadline miss pattern, which needs to be obtained through schedulability analysis under weakly-hard constraints. Such analysis (e.g., for evaluating $dmm_i(N_i^j)$ in Equation (13)) can hardly be captured with closed-form equations and addressed by existing solvers. The exploration of the feasible weakly-hard constraints under both control and schedulability analysis further increases the complexity. Thus, we developed a meta-heuristic algorithm to solve the problem. The algorithm first uses a heuristic method based on bin packing for generating an initial configuration of the security monitoring tasks, and then carries out a simulated annealing (SA) process to explore the system configuration space via random permutation of the allocation, priority, and period of the security monitoring tasks. During the SA process, for each system configuration, the algorithm calls routines for schedulability analysis, control analysis, and evaluation of security constraints and objective. This provides the feasibility of various constraints and the overall objective value (Equation (15)). The algorithm then decides whether to accept or reject the randomly generated configuration, and continues the SA process until it ends.

Next, we will introduce our schedulability analysis method, the initial solution generation, and the overall algorithm. The control analysis and security evaluation follow Section III.

A. Schedulability Analysis

An important element of our algorithm is the schedulability analysis under weakly-hard constraints (Algorithm 1), which analyzes the scheduling feasibility and the control task deadline miss patterns for a given system configuration. As stated before, we assume that a task will be killed the moment it misses its deadline (similar strategy as discussed in [14]). The analysis includes two steps. First, a worst-case response time (WCRT) analysis based on busy window analysis [23] is performed to check whether there are any deadline misses (line 1 to 3). If there is not any, the system is already schedulable. Otherwise, an event-based simulation (line 4 to 27) calculates the deadline miss pattern for each task within the hyper-period. One round of such simulation can derive the deadline miss patterns for all tasks[4].

The event-based simulation simulates the execution order of each task by recording the time-stamp of each event, including job release, completion, etc. $\theta_{ij} = (s_{\theta_{ij}}, c_{\theta_{ij}})$ is the j-th job of task τ_i, where $s_{\theta_{ij}} = j \cdot t_{\tau_i}$ is the release time of the job and $c_{\theta_{ij}}$ is the remaining computation time of the job. $Miss[i][j] = true$ if τ_i's j-th job θ_{ij} misses its deadline. $event\_queue$ and job_queue are two job priority queues for unreleased jobs and for released but unfinished jobs, respectively. After the event-based simulation, function $VerifyWHConstraint()$ verifies weakly-hard constraints by counting the number of deadline misses within any consecutive N_i^j activations. The system schedulability and deadline miss pattern $Miss$ will be returned at the end of the algorithm. A detailed description of Algorithm 1 can be found in Appendix B.

B. Initial Solution Generation

Given the system profile of a set of control tasks and other tasks allocated and scheduled on ECUs (and messages transmitted on a CAN bus), the first step in our algorithm is to generate an initial solution for the simulated annealing process. We use a bin packing based method for this step. For each critical CAN message m_i to be monitored, we iteratively find μ_i ECUs with the lowest utilization and add corresponding security monitoring tasks, where μ_i is the given redundancy level. After security tasks are added to monitor all the critical CAN messages (while satisfying their coverage and redundancy requirements), we check the system schedulability with $CheckSched()$. If the system is unschedulable, we will identify all the security tasks with deadline misses and increase their periods. The initialization result may still be unschedulable at the end, which will be further addressed in the simulated annealing process. A detailed description for the initialization and the pseudo code can be found in Appendix C.

C. Overall Algorithm

Algorithm 2 shows our overall optimization algorithm. First, an initial solution is generated as introduced above

[4]The computational complexity of the event-based simulation depends on the total number of events within a hyper-period. In our preliminary experiments of 25 tasks, with periods ranging from 50 to 1000 ms, our event-based simulation approach is 20 times faster than the analysis method in [19].

Algorithm 1: $CheckSched()$: Schedulability Analysis

1: $WCRTAnalysis(\mathcal{T})$
2: **if** $\forall \tau_i \in \mathcal{T}$, $r_{\tau_i} \leq d_{\tau_i}$ **then**
3: **return true**
4: **for** task $\tau_i \in \mathcal{T}$ **do**
5: $event\_queue.push(\theta_{i0})$
6: $\theta_{ij} = event\_queue.pop(), \quad cur\_time = s_{\theta_{ij}}$
7: $event\_queue.push(\theta_{i(j+1)})$
8: **while** $cur\_time \leq HyperPeriod$ **do**
9: **while** $s_{\theta_{ij}} \leq cur\_time$ **do**
10: $job\_queue.push(\theta_{ij})$
11: $\theta_{ij} = event\_queue.pop()$
12: $event\_queue.push(\theta_{i(j+1)})$
13: **if** job_queue is empty **then**
14: $cur\_time = s_{\theta_{ij}}$
15: **else**
16: $\theta_{kl} = job\_queue.pop(), \quad next = s_{\theta_{ij}}$
17: $response = cur\_time + c_{\theta_{kl}}$
18: **if** $s_{\theta_{kl}} + d_{\tau_k} < next$ **or** $response \leq next$ **then**
19: **if** $response \leq s_{\theta_{kl}} + d_{\tau_k}$ **then**
20: $Miss[k][l] =$ **false**
21: $cur\_time = response$
22: **else**
23: $Miss[k][l] =$ **true**
24: $cur\_time = \max(s_{\theta_{kl}} + d_{\tau_k}, cur\_time)$
25: **else**
26: $c_{\theta_{kl}} = c_{\theta_{kl}} - (next - cur\_time)$
27: $job\_queue.push(\theta_{kl})$, $cur\_time = next$
28: $schedulability = VerifyWHConstraint(Miss)$
29: **return** $Miss, schedulability$

Algorithm 2: Our Optimization Algorithm

1: $S_0 = Initialization()$
2: $S_{best} = S_{cur} = S_{new} = S_0$
3: $\eta_{best} = \eta_{cur} = \eta_{new} = ComputeObj(S_0)$
4: **while** $T > T^*$ **do**
5: $k = 1$
6: **while** $k \leq iter\_max$ **do**
7: $S_{new} = RandomMove(S_{cur})$
8: $miss, is\_sched = CheckSched(S_{new})$
9: $ctrl\_obj, is\_stable = ComputeCtrl(miss)$
10: $sec\_obj, sec\_feasible = ComputeSec(S_{new})$
11: $\eta_{new} = \alpha * sec\_obj + \beta * ctrl\_obj$
12: **if** $is\_sched \wedge is\_stable \wedge sec\_feasible ==$ **false** **then**
13: $\eta_{new} = \eta_{new} + ComputePenalty(S_{new})$
14: **if** $\eta_{new} < \eta_{cur}$ **then**
15: $S_{cur} = S_{new}, \eta_{cur} = \eta_{new}$
16: **if** $is\_sched \wedge is\_stable \wedge sec\_feasible ==$ **true** **then**
17: $S_{best} = min(S_{cur}, S_{best})$
18: $\eta_{best} = min(\eta_{cur}, \eta_{best})$
19: **else if** $AccepProb(\eta_{new} - \eta_{cur}, T) > rand()$ **then**
20: $S_{cur} = S_{new}, \eta_{cur} = \eta_{new}$
21: $k = k + 1$
22: $T = T * cooling\_factor$
23: $MergeMonitoringTasks()$
24: **return** S_{best}, η_{best}

(line 1). Then, a simulated annealing process is conducted to explore the design space. During each step of the simulated annealing, the current system configuration S_{cur} is randomly changed to generate a new configuration S_{new} in the function $RandomMove()$ (line 7). The random move could be changing the allocation of a security monitoring task to another ECU, swapping the priorities between a security monitoring task and a control task or an other task on the same ECU, or changing the period of a security monitoring task.

Once a new configuration is presented, the algorithm calls the schedulability analysis routine $CheckSched()$ (Algorithm 1) to evaluate the schedulability and obtain control task deadline miss patterns. With the derived miss patterns, the algorithm calls a routine $ComputeCtrl()$ to evaluate control stability and performance, as introduced in Section III-A. It also calls a routine $ComputeSec()$ to evaluate security objective and constraints, as defined in Section III-B. The overall objective value is then computed as defined in Equation (15) (line 11). If the new configuration fails to satisfy the schedulability, security constraints or control stability, a penalty will be added to the overall objective value based on the degree of constraint violations in the function $ComputePenalty()$ (line 13). We accept the new configuration if the new objective η_{new} is better than previous solution; otherwise, the acceptance probability is calculated based on current temperature and the objective difference.

After the simulated annealing process completes, we merge the security monitoring tasks that have the same period and are on the same ECU in the function $MergeMonitoringTasks()$ (line 23). This will not affect the objective function or constraints in our current formulation, however could help reduce the switching overhead in practice.

V. Experimental Results

We evaluate our weakly-hard based codesign method with an industrial case study and a set of synthetic examples. In these experiments, we derive the controllers based on the example LTI systems from [15], [43], [44], [45].

A. Industrial Case Study

We first conduct experiments on an industrial subsystem of an experiment vehicle (derived from the one in literature [46]). The experiment vehicle supports data collection from 360° sensors and sending control signals to actuators, such as brake, throttle, steering, etc. There are in total 41 tasks with given periods and WCETs, distributed on 9 ECUs. 83 messages are transmitted through a CAN bus. The periods of tasks and messages range from 10 to 100 ms. The desired period of each security monitoring task is set to be the period of the corresponding critical messages it monitors while $T_{i,j}^{max} = 4 \times T_{i,j}^{des}$.

Weakly-hard vs. Hard: We first compare the security objective value between our weakly-hard based approach and the

traditional system with only hard deadlines. Table I shows the results of the two cases when different number of messages are deemed as critical and need to be monitored. The weakly-hard results are obtained by running our algorithm (Algorithm 2) with the objective function in Equation (15) only including security (i.e., $\beta = 0$). The hard deadline results are obtained by running our algorithm with only security objective and hard deadlines. As we can see, allowing weakly-hard constraints (while ensuring control stability and performance requirements) can significantly improve the system's capability to accommodate security monitoring tasks. Note that when the number of monitored message is 50 or 60, only hard deadlines cannot yield feasible solution (i.e., some messages cannot be monitored with meaningful efficacy).

TABLE I
COMPARISON ON SECURITY BETWEEN ALLOWING WEAKLY-HARD CONSTRAINTS AND ONLY HARD DEADLINES FOR INDUSTRIAL EXAMPLE.

# of critical messages	20	30	40	50	60
security metric (weakly-hard)	1.42	1.85	2.22	2.66	2.85
security metric (hard)	1.70	2.38	2.93	n.a.	n.a.

Trading off control performance and security: Leveraging weakly-hard constraints to improve security is at the expense of degraded control performance. We conduct experiments to quantitatively evaluate such tradeoff. Fig. 2 shows the pseudo Pareto front between control performance metric (Equation (9)) and security metric (Equation (12)) when different weights of α and β are chosen in the overall objective function (Equation (15)) during our optimization. 50 critical messages are selected to be monitored with a redundancy level of 2. The tradeoff trend is very clear and the results demonstrate the effectiveness of our codesign approach in addressing the two objectives via weakly-hard constraints.

Moreover, two boundary points are also shown in Fig. 2. The red asterisk point at bottom-right corresponds to a system configuration without any deadline miss, i.e., lower bound for control performance in our model. However, it cannot satisfy the security monitoring constraints as defined in (11). The red cross point at top-left corresponds to a system configuration with all security monitoring tasks having the desired period, i.e., lower bound for security metric. However, it violates the control stability constraints as there are too many deadline misses. These two points demonstrate the necessity to address control and security in a codesign approach.

B. Synthetic Examples

We also conduct experiments with a set of synthetic examples of 30 tasks on 4 ECUs. The examples have varying initial system utilization before adding monitoring tasks (from 0.6 to 0.9 on each ECU). 21 CAN messages are selected to be monitored, with a redundancy level of 2. The allocation for the control and other tasks is decided based on a bin packing approach and their priorities are based on rate monotonic policy. Each point in Fig. 3 is the average result of 20 synthetic examples. The figure clearly demonstrates the capability of our codesign method to trade off control performance and security

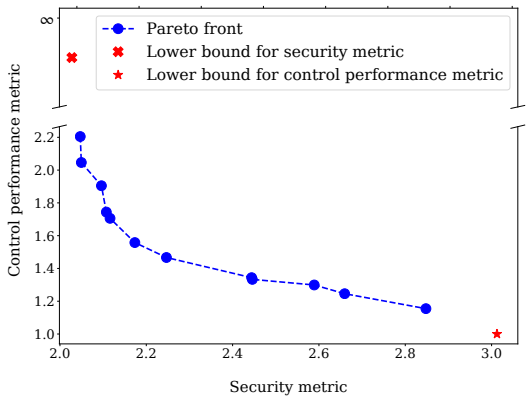

Fig. 2. Tradeoff between control and security for industrial example.

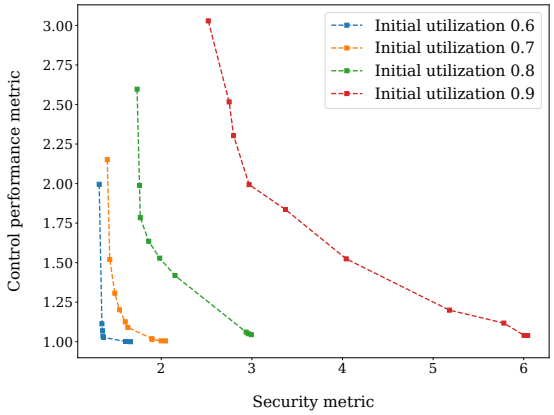

Fig. 3. Tradeoff between control performance and security for synthetic examples under different initial utilizations.

(by selecting different weights for the two objectives in our optimization), similar to the industrial example. Furthermore, we can observe that under relatively low utilization, it is easier to improve security (i.e., shortening periods of security monitoring tasks) without substantial impact on control performance, while under higher utilization, such action will have more significant impact on control. This trend is reasonable, and the quantitative results could facilitate design decisions under different utilizations.

VI. CONCLUSION

We presented a codesign approach to leverage weakly-hard constraints for improving system security while considering control performance and stability. Our approach explores the allocation, priority, and period assignment of security monitoring tasks to optimize a joint objective function of security and control performance, while meeting requirements on control stability, schedulability, and security. Experimental results demonstrate the significant potential of leveraging weakly-hard constraints and the effectiveness of our approach to fully realize such potential.

VII. ACKNOWLEDGEMENT

We gratefully acknowledge the support from the US National Science Foundation awards 1834701, 1834324, 1839511, and 1724341; and Imprint India Project No. 6158.

APPENDIX

A. Example of the System Model

As an example of our system model, in Fig. 1, two control tasks τ_5 and τ_6 are allocated to ECU1 and ECU3, respectively. A security monitoring task τ_7 is deployed on ECU4. τ_1, τ_2, τ_3 and τ_4 are tasks implementing other functions. Assuming an intruder gains access to the system and takes control of ECU2, it may spoof a benign message stream m, and send its spoofed message stream $\bar{m}$ at a higher frequency in order to suppress the original message m. Security monitoring task τ_7 can leverage specific knowledge of the system to detect the abnormal message stream $\bar{m}$ and inform system about the intrusion, as discussed in [7].

B. Event-based Simulation for Schedulability Analysis

This section presents a more detailed description of our schedulability analysis method under weakly-hard constraints, as introduced before in Section IV-A.

In Algorithm 1, we denote the j-th invocation of task τ_i as job θ_{ij}, which can be represented by a tuple $(s_{\theta_{ij}}, c_{\theta_{ij}})$. Here, $s_{\theta_{ij}} = j \cdot t_{\tau_i}$ is the release time of the job, and $c_{\theta_{ij}}$ is the remaining computation time of the job, which is initially set as $c_{\theta_{ij}} = c_{\tau_i}$. For jobs of task τ_i, their deadline miss patterns are recorded in an array $Miss[i]$, where $Miss[i][j] = true$ if τ_i's j-th job θ_{ij} misses its deadline. During the simulation, $event\_queue$ and job_queue are two job priority queues to store the unreleased jobs and released but unfinished jobs, respectively. While $event\_queue$ is sorted by the job release time $s_{\theta_{ij}}$, job_queue is sorted by the task priority.

At current time point cur_time, any jobs that can be released are popped from the $event\_queue$ and pushed into the job_queue, and the highest priority job in the job_queue is scheduled to run. Here, θ_{kl} is the scheduled job at cur_time and θ_{ij} is the next job to release. Then, the simulation moves to the next time point (the scheduled job's deadline $s_{\theta_{kl}} + d_{\tau_k}$, the scheduled job's response time $response$, or the next job's release time $next = s_{\theta_{ij}}$). In this work, we assume that if a job misses its deadline, it will be killed and no further computation is needed. If the scheduled job has not finished (or been killed) at time $next$, it will update its remaining execution time $c_{\theta_{ij}}$ and be pushed back to the job_queue. Every time a job θ_{kl} finishes (or gets killed), the simulation records whether it misses its deadline in $Miss[k][l]$. After the simulation completes, the function $VerifyWHConstraint()$ counts the maximum deadline misses of any consecutive N_i^j activations (i.e., $dmm_i(N_i^j)$) to verify whether tasks with weakly-hard constraints meet those constraints. The return value $schedulability$ indicates the overall schedulability, and $Miss$ represents the deadline miss patterns.

C. Initial Solution Generation

This section presents a more detailed description of our initial solution generation method, as introduced before in Section IV-B.

The pseudo code for our initial system generation method is shown in Algorithm 3. For each CAN message m_i, we iteratively find μ_i ECUs and add security tasks on those ECUs to monitor this message (line 1 to 7). Here μ_i is the redundancy level requirement for m_i, i.e., m_i has to be monitored by μ_i security tasks on different ECUs, as defined in Section III-B. More specifically, in each iteration, we find the ECU e_j currently with the lowest utilization, add a security monitoring task on e_j for m_i, and then remove e_j from $\mathcal{E}$. Function $AddMonitoringTask()$ also updates the utilization of e_j and sets the initial period of the new security task the same as m_j's period.

After security tasks are added to monitor all the critical CAN messages (while satisfying their coverage and redundancy requirements), we check the system schedulability with $CheckSched()$. If the system is not schedulable (very likely in practice), we will identify all the security tasks that have deadline misses and scale their periods by an integer factor s, until the system becomes schedulable or s reaches K_{max} (line 8 to 10).

Algorithm 3: $Initialization()$: Initial Solution Generation

```
1:  for each message m_i in M_cri do
2:      k = μ_i, E = {e_1, e_2, ..., e_{n_e}}
3:      while k > 0 do
4:          Choose ECU e_j with the lowest utilization among E
5:          a_{m_i,e_j} = 1, k = k − 1
6:          AddMonitoringTask(m_i, e_j)
7:          remove e_j from E
8:  for s ← 2 to K_max do
9:      if CheckSched() == false then
10:         Scale the periods of deadline-missed security tasks
            by s
```

REFERENCES

[1] K. Koscher, A. Czeskis, F. Roesner *et al.*, "Experimental security analysis of a modern automobile," in *IEEE Symposium on Security and Privacy*, 2010.

[2] S. Checkoway, D. McCoy, B. Kantor *et al.*, "Comprehensive experimental analyses of automotive attack surfaces," in *USENIX Security Symposium*, 2011.

[3] C. Miller and C. Valasek, "Remote exploitation of an unaltered passenger vehicle," *Black Hat USA*, 2015.

[4] D. Nilsson, U. Larson, and E. Jonsson, "Efficient in-vehicle delayed data authentication based on compound message authentication codes," in *VTC*, 2018.

[5] B. Groza, S. Murvay, A. van Herrewege *et al.*, "Libra-can: a lightweight broadcast authentication protocol for controller area networks," in *CANS*, 2012.

[6] C. Lin, B. Zheng, Q. Zhu *et al.*, "Security-aware design methodology and optimization for automotive systems," *ACM TODAES*, vol. 21, no. 1, 2015.

[7] P. Waszecki, P. Mundhenk, S. Steinhorst *et al.*, "Automotive electrical and electronic architecture security via distributed in-vehicle traffic monitoring," *IEEE TCADICS*, vol. 36, no. 11, pp. 1790–1803, 2017.

[8] K. Cho and K. G. Shin, "Fingerprinting electronic control units for vehicle intrusion detection," in *USENIX Security Symposium*, 2016.

[9] H. M. Song, H. R. Kim, and H. K. Kim, "Intrusion detection system based on the analysis of time intervals of can messages for in-vehicle network," in *ICOIN*, 2016.

[10] M. Müter and N. Asaj, "Entropy-based anomaly detection for in-vehicle networks," in *IEEE Intelligent Vehicles Symposium (IV)*, 2011.

[11] C. Lin, Q. Zhu, and A. Sangiovanni-Vincentelli, "Security-aware mapping for TDMA-based real-time distributed systems," in *Computer-Aided Design (ICCAD), 2014 IEEE/ACM International Conference on*, Nov 2014, pp. 24–31.

[12] C. Lin, Q. Zhu, C. Phung, and A. Sangiovanni-Vincentelli, "Security-aware mapping for CAN-based real-time distributed automotive systems," in *Computer-Aided Design (ICCAD), 2013 IEEE/ACM International Conference on*, 2013, pp. 115–121.

[13] M. Hasan, S. Mohan, R. B. Bobba *et al.*, "Exploring opportunistic execution for integrating security into legacy hard real-time systems," in *RTSS*, 2016.

[14] P. Pazzaglia, L. Pannocchi, A. Biondi *et al.*, "Beyond the weakly hard model: measuring the performance cost of deadline misses," in *ECRTS*, 2018.

[15] D. Goswami, R. Schneider, and S. Chakraborty, "Relaxing signal delay constraints in distributed embedded controllers," *IEEE Transactions on Control Systems Technology*, vol. 22, no. 6, pp. 2337–2345, 2014.

[16] E. P. van Horssen, A. R. B. Behrouzian, D. Goswami *et al.*, "Performance analysis and controller improvement for linear systems with (m, k)-firm data losses," in *European Control Conference (ECC)*, 2016.

[17] M. B. Gaid, D. Simon, and O. Sename, "A design methodology for weakly-hard real-time control," 2008, 17th IFAC World Congress.

[18] G. Frehse, A. Hamann, S. Quinton *et al.*, "Formal analysis of timing effects on closed-loop properties of control software," in *RTSS*, Dec 2014.

[19] G. Bernat, A. Burns, and A. Liamosi, "Weakly hard real-time systems," *IEEE Transactions on Computers*, vol. 50, no. 4, pp. 308–321, 2001.

[20] P. Ramanathan, "Overload management in real-time control applications using (m, k)-firm guarantee," *IEEE TPDS*, 1999.

[21] C. Huang, W. Li, and Q. Zhu, "Formal verification of weakly-hard systems," in *Proceedings of the 22Nd ACM International Conference on Hybrid Systems: Computation and Control*, ser. HSCC '19. New York, NY, USA: ACM, 2019, pp. 197–207. [Online]. Available: http://doi.acm.org/10.1145/3302504.3311811

[22] C. Huang, K. Wardega, W. Li, and Q. Zhu, "Exploring weakly-hard paradigm for networked systems," in *Workshop on Design Automation for CPS and IoT (DESTION'19)*, 2019.

[23] S. Quinton, M. Hanke, and R. Ernst, "Formal analysis of sporadic overload in real-time systems," in *DATE*, 2012.

[24] W. Xu, Z. A. Hammadeh, A. Kroller *et al.*, "Improved deadline miss models for real-time systems using typical worst-case analysis," in *ECRTS*, 2015.

[25] Y. Sun and M. D. Natale, "Weakly hard schedulability analysis for fixed priority scheduling of periodic real-time tasks," *ACM TECS*, vol. 16, no. 5s, 2017.

[26] L. Ahrendts, S. Quinton, T. Boroske *et al.*, "Verifying weakly-hard real-time properties of traffic streams in switched networks," in *ECRTS*, 2018.

[27] M. Shirazi, M. Kargahi, and L. Thiele, "Resilient scheduling of energy-variable weakly-hard real-time systems," in *Proceedings of the 25th International Conference on Real-Time Networks and Systems*, ser. RTNS '17. New York, NY, USA: ACM, 2017, pp. 297–306.

[28] L. Niu and D. Zhu, "Reliability-aware scheduling for reducing system-wide energy consumption for weakly hard real-time systems," *Journal of Systems Architecture*, vol. 78, pp. 30 – 54, 2017. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1383762116301825

[29] B. Zheng, P. Deng, R. Anguluri, Q. Zhu, and F. Pasqualetti, "Cross-layer codesign for secure cyber-physical systems," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 35, no. 5, pp. 699–711, May 2016.

[30] P. Deng, Q. Zhu, A. Davare, A. Mourikis, X. Liu, and M. D. Natale, "An efficient control-driven period optimization algorithm for distributed real-time systems," *IEEE Transactions on Computers*, vol. 65, no. 12, pp. 3552–3566, December 2016.

[31] Q. Zhu and A. Sangiovanni-Vincentelli, "Codesign methodologies and tools for cyberphysical systems," *Proceedings of the IEEE*, vol. 106, no. 9, pp. 1484–1500, Sep. 2018.

[32] U. Eklund and H. Gustavsson, "Architecting automotive product lines: industrial practice," *Science of Computer Programming*, vol. 78, no. 12, pp. 2347–2359, 2013.

[33] Q. Zhu, Y. Yang, M. D. Natale, E. Scholte, and A. Sangiovanni-Vincentelli, "Optimizing the software architecture for extensibility in Hard real-time distributed systems," *the IEEE Transactions on Industrial Informatics*, vol. 6, no. 4, pp. 621–636, 2010.

[34] J. Axelsson, "Evolutionary architecting of embedded automotive product lines: an industrial case study," in *Software Architecture, 2009 & European Conference on Software Architecture. WICSA/ECSA 2009. Joint Working IEEE/IFIP Conference on*. IEEE, 2009, pp. 101–110.

[35] M. Müter, A. Groll, and F. C. Freiling, "A structured approach to anomaly detection for in-vehicle networks," in *IEEE IAS*, 2010.

[36] U. E. Larson, D. K. Nilsson, and E. Jonsson, "An approach to specification-based attack detection for in-vehicle networks," in *IEEE Intelligent Vehicles Symposium*, 2008.

[37] P. Waszecki, M. Lukasiewycz, and S. Chakraborty, "Decentralized diagnosis of permanent faults in automotive e/e architectures," in *SAMOS*, 2015.

[38] T. A. Henzinger, B. Horowitz, and C. M. Kirsch, "Giotto: a time-triggered language for embedded programming," *Proceedings of the IEEE*, vol. 91, no. 1, pp. 84–99, Jan 2003.

[39] K. J.Åström and B.Wittenmark, *Computer-controlled systems: Theory and Design*. Prentice Hall, 1997.

[40] W. Chang, L. Zhang, D. Roy, and S. Chakraborty, *Control/architecture codesign for cyber-physical systems*, ser. Handbook of Hardware/Software Codesign. Springer Netherlands, 2017.

[41] J.Kautsky, N. K. Nichols, and P. Van Dooren, "Robust pole assignment in linear state feedback," *International Journal of Control*, vol. 41, no. 5, 1985.

[42] M. Hasan, S. Mohan, R. Pellizzoni, and R. B. Bobba, "Contego: An adaptive framework for integrating security tasks in real-time systems," *arXiv preprint arXiv:1705.00138*, 2017.

[43] W. C. Messner, D. M. Tilbury, and A. P. R. Hill, "Control tutorials for matlab® and simulink®," 1999.

[44] J. Vijayshre, "Position control of dc motor using genetic algorithm based pid controller." *International Journal of ElectroComputational World & Knowledge Interface*, vol. 1, no. 2, 2011.

[45] W. Chang, A. Prbstl, D. Goswami *et al.*, "Battery- and aging-aware embedded control systems for electric vehicles," in *RTSS*, Dec 2014.

[46] Q. Zhu, H. Zeng, W. Zheng *et al.*, "Optimization of task allocation and priority assignment in hard real-time distributed systems," *ACM Trans. Embed. Comput. Syst.*, vol. 11, no. 4, pp. 85:1–85:30, Jan. 2013.