CRII: CNS: Integrating Security Tasks into Multicore Real-Time Systems

A Proposal to the
National Science Foundation
September 5, 2021
Monowar Hasan, Wichita State University

Overview

Embedded real-time systems (RTS) are all around us — they are used to monitor and control physical systems and processes in various domains such as unmanned vehicles, self-driving cars, critical infrastructures, manufacturing systems, medical devices to name but a few. Until recently, security was not an important concern in the design of such systems since they used proprietary protocols, platforms and software, and were not connected to the outside world. Recent attacks though have shown that such critical systems are vulnerable to malicious attacks, and it can result in serious damage to humans, the systems as well the environment. Integrating security, in such systems is not an easy task since RTS have stringent timing and safety constraints — hence a security mechanism must not violate these requirements. There has also been a lot of interest in using multicore processors in RTS and these open up interesting security challenges as well. ~~In this~~ The proposal, we (a) ~~aim~~ to ~~Study~~ ies and develop s frameworks for integrating security into RTS built using multicore processors, ~~As part of this proposed work, we will study~~ (b) analyze the various parameters that affect design choices while integrating security into multicore RTS, ~~We will~~ and (c) ~~also~~ devise metrics that will enable ~~us~~ the designers to measure success and objectively analyze different solutions. ~~We will~~ The Proposed Solutions wills be evaluate evaluate our solutions on a variety of platforms — from simulation engines to real hardware (automated rovers and a scale-down robot testbed).

Intellectual Merit

~~Our~~ Thesis proposal addresses a fundamentally challenging problem, i.e., how do we integrate security monitoring techniques into RTS without affecting timing requirements? The focus of ~~our~~ ~~the~~ the proposed research is to develop techniques that will allow us to trade back and forth between these, seemingly, conflicting properties so that designers of the systems can correctly gauge system requirements —– hence, perhaps, meeting both, the real-time constraints as well as integrating effective security mechanisms. The proposed work will bridge the knowledge gap by analyzing the trade-offs between the two requirements — whether reducing the promises made to one (say a reduction of the security goals) will result in improvements for the other (better real-time guarantees). Designers of such systems will then have the ability to make well-informed choices and develop systems that are inherently safer and more secure.

Broader Impacts

The proposed research has far-reaching societal, scientific, and educational impacts. The development of analysis techniques and system-level frameworks proposed in this work will inherently make critical systems of modern society (such as aircraft, automobiles, power grid, unmanned vehicles, satellites, manufacturing plants, industrial control systems, medical devices, and critical infrastructures) more secure, and hence, safer. The proposed work can lead to substantial improvements in consumer cyber-physical products and improve systems that have national security considerations. The proposed educational and outreach activities are expected to have a broad and positive societal impact by training a qualified, diverse workforce. Specifically, the education efforts include *(a)* development of a new senior undergraduate/graduate-level course on Real-Time Systems and *(b)* enhancement of existing Introduction to Cybersecurity course with advanced topics on real-time scheduling and security as well as results generated from this research. The PI will also actively work on broadening participation in computing by *(a)* providing research opportunities for the students from underrepresented communities through the Wichita State's Shocker Engineering Academy (SEA) program, *(b)* arranging security boot camps for high school students through Wichita State's security education hub (HCEA), *(c)* mentoring minority undergraduate student groups through Computing Research Association's Distributed Research Experiences for Undergraduates (DREU) initiatives, and *(d)* volunteering for presentations/demos at the events targeting K12 students (e.g., National Security Agency's K12 CyberTalk) to empower them pursuing a career in cyber-physical systems and cybersecurity.

1 Introduction

Real-time systems (RTS) systems such as avionics, nuclear power plants, automobiles, space vehicles, power generation and distribution systems, medical devices, industrial robots have been in existence for decades and play a vital role in shaping today's technological evolution from everyday living to space exploration. Such systems have *safety-critical* properties and *stringent timing* requirements. Any problems in RTS could result in significant harm to humans, the system, or even the environment. Systems with real-time requirements need to function correctly, but *within their predefined timing constraints*, often termed as "deadlines". For example, consider real-time applications with stringent timing constraints such as deployment of a car's airbag or an industrial robot operates on a manufacturing conveyor line. A typical window for an airbag deployment (time between detection of collision and final airbag operations) is around 50–60 ms [1] and the response time for robot movements (i.e., placing and moving objects on the conveyor) is around 50–100 ms [2].

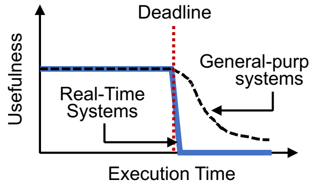

Figure 1: Timeliness requirements of RTS.

If real-time applications failed to comply with their timing requirements (deadlines), the usefulness of results produced by the system drops sharply (see Figure 1). From the earlier airbag deployment and manufacturing robot example, if the application tasks fail to deploy the airbag in time or there is a delay to update the angle of rotation of the robot arm, the physical system may not work properly and hence put the safety of the human operators at risk. This is different from general-purpose systems where the usefulness drops in a more gradual manner (e.g., a web service may tolerate a few millisecond delay without degrading user experience significantly).

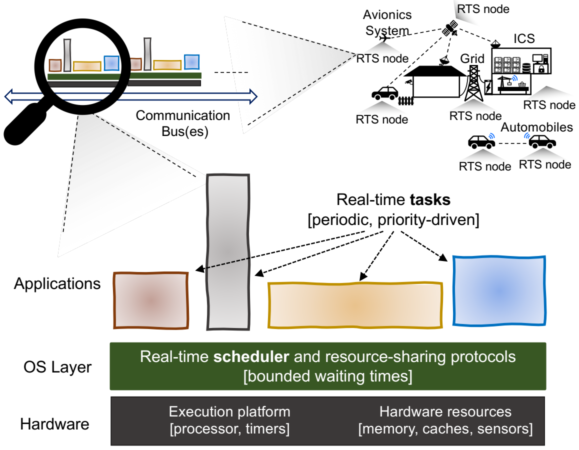

Figure 2: Abstraction of a real-time node with common uses cases.

Figure 2 presents a high-level illustration of a RTS. Each real-time application in the system (called "task") represents a time-critical function, and a collection of such tasks are hosted on a hardware platform. The scheduler in real-time operating system (RTOS) uses timers and interrupt handlers to enforce timing guarantees at runtime. Access to shared platform resources (such as caches, buses, memory) is regulated using sophisticated resource sharing protocols to ensure that deadlines can be met.

Although safety and fault-tolerance have long been important design focus in such systems, security has rarely been a consideration in the design of RTS mainly due to beliefs such as: *(a)* RTS lack inherent value to adversaries (*"why would anyone attack them?"*), *(b)* the prevalence of custom hardware/software/protocols will deter attackers (*"these protocols/hardware/software are secret and so arcane that no one can decipher them"*), and also *(c)* the lack of computing power and memory in these systems will throttle potential adversarial actions (*"what can they do even if they get in?"*). While traditionally RTS adopted proprietary protocols, platforms, software and were air-gapped (i.e., not connected to the outside world), with the advent of newer domains such as autonomous vehicles, drones, remote monitoring and control, and Internet-of-Things (IoT)-specific applications, RTS find themselves front and center in modern society. Since many RTS now use commodity-off-the-shelf (COTS) components and are often connected to each other or even the Internet, they exposed additional attack surfaces, often overturning all of the aforementioned beliefs. A number of high-profile attacks on industrial control systems (e.g., Stuxnet [3], BlackEnergy [4]), attack demonstrations by researchers on automobiles [5,6], avionic systems [7], drones [8,9], and medical devices [10] have shown that the threat is real and systems ~~composed of RTS might be~~ vulnerable to cyber attacks. Given the

are

such are

increasing security risks, it is essential to have a layered defense and integrate resilience against cyber attacks into the design of embedded RTS. However, stringent timing constraints severely inhibit how security solutions can be added to RTS; for instance, the *protection methods should not cause any timing problems*.

Research Questions. Integrating security in RTS is not straightforward since monitoring/detection mechanisms must *(a)* co-execute with existing real-time tasks, *(b)* comply with timing/safety constraints, and *(c)* designed/scheduled in a way that an adversary cannot easily evade them. For instance, *how often and how long should a monitoring and detection task run to be effective but not interfere with real-time, control, or other safety-critical tasks?* How do we execute monitoring tasks in a *tamper-proof manner* and ensure their functional correctness? The challenge is then, ~~how do we~~ is to develop solutions that address the apparent tension between *security requirements* (i.e., having enough cycles and resilience for effective monitoring and detection) and the *timing and safety requirements* (i.e., not interfere with deadlines)? Although RTS have traditionally been designed using single-core chips, *multicore platforms* are becoming increasingly common since they provide better performance and energy efficiency [11, 12]. Security integration techniques for multicore RTS even more non-trivial in particular when considering that *(a)* isolation in multicore systems is a challenge itself [12–14] and *(b)* designers have multiple choices across cores to retrofit security mechanisms. As an example, should we *(a) statically assign processor cores* for security monitoring or *(b)* allow *runtime migration* (for better detection performance)? For the latter, *how do we ensure that the temporal guarantees of critical real-time tasks are not affected by runtime migration*?

Hence, our proposal focuses on addressing the following research questions.

- How do we *integrate* and then *characterize the effects of security* in RTS especially those designed using *multicore* chips?
- What are the *trade-offs* on the security and timing requirements while guaranteeing *no (or minimal) perturbations* for the real-time properties?
- What are the *performance criteria and metrics* that need to be considered?

In answer to the above research questions, we will ***(a)* devise *novel algorithms, scheduling models, and frameworks* to integrate security** into multicore platforms **that are cognizant with real-time requirements, *(b)* build *design-time tools and system-level plugins* to incorporate our proposed techniques into off-the-shelf systems, and *(c)* develop *metrics* to carefully trade-off two contending requirements: timeliness and security**.

1.1 Proposed Research

In this project we will develop an unified framework that will *integrate monitoring and detection mechanisms into **multicore** RTS* (Section 4). Our proposed research will show that *it is possible to integrate security into RTS by a careful scheduler-level analysis of, and co-design with, system constraints, viz. software, hardware, and timing requirements*. Our techniques will assist the designers of systems to better understand *(a)* how to *integrate security* into multicore RTS and *(b)* what are the *trade-offs* on the security front while guaranteeing minimal (or no) perturbations for the real-time properties. Our end goal, then, is to *provide designers with a knob that they can use to tune to one side or the other — real-time vs. security*.

Security Task	**Approach/Tools**
File-system checking	Tripwire [15], AIDE [16]
Network packet monitoring	Bro [17], Snort [18]
Hardware event monitoring	perf [19], OProfile [20]
Application specific checks	Behavior-based detection [21–24]

Table 1: Example of security tasks — this is by no stretch meant to be an exhaustive list.

When integrating any security mechanisms into RTS, the designers need to ensure that they do not perturb or impact the real-time functions in any significant way while at the same time provide the necessary level of security. This is especially acute in the case of *legacy systems* (i.e., existing RTS where modification or perturbation of task parameters such as run-times, periods, and task execution orders is not always feasible.). Hence, we propose to improve the security posture of RTS through integration of ***"security tasks"*** (i.e., tasks that are specific for intrusion monitoring and detection purposes) while ensuring that the existing real-time tasks are *not affected* by such integration. Table 1 presents some examples of security tasks that can be integrated into l~~egacy syste~~ms RTS. Security tasks could include protection, detection or response mechanisms, depending on the system requirements — for instance, a sensor correlation task (to detect sensor manipulation) or an anomaly detection task (that checks possible intrusions) [25]. Our proposed work will *not* design towards any specific security mechanisms — our focus here is to *integrate any designer-provided security monitoring tasks* and *schedule* them with respect to real-time constraints into a multicore RTS.

We will evaluate our proposed techniques using *(a)* simulations, *(b)* realistic workloads and embedded benchmark suites, and *(c)* off-the-shelf hardware testbeds (Section 5). Our research activities will be complemented with tightly integrated educational components reaching from undergraduate/graduate students, students from minority communities, and K-12 students (Section 7). The main outcomes of our research will be to:

- understand the various parameters that affect design choices while integrating security into RTS;
- develop new analytical frameworks and metrics to integrate security monitoring mechanisms into multicore RTS; and
- integrate proposed frameworks into ~~existing real-tim~~e commodity OS kernels an~~d trusted enclave~~s and evaluate design trade-offs.

Intellectual Merit. Our proposed research will develop a foundational body of work to improve the security of RTS especially those built using multicore chips. It will tackle a fundamentally complex problem: *how to integrate monitoring and detection techniques in RTS while still meeting the stringent timing constraints imposed upon such systems?* By systematically integrating security monitoring tasks, the proposed work should *increase the difficulty for adversaries*. Finally, the proposed work also addresses another critical need: *how do we measure/quantify the effects of security and analyze trade-offs between security and real-time guarantees?*

1.2 Justification for Funding Request

Foundations for Long-Term Research. The activities proposed here are critical steps for the PI to launch his research career. The proposed work will provide solid foundations for developing security techniques in diverse domains — from RTS to broader cyber-physical, IoT/edge systems, and even general-purpose computing platforms. While the immediate focus of this research is on integrating security monitoring mechanisms info RTS; the PI believes that the ideas developed in this work can be extended to more general-purpose computing systems and will serve as the basis for future competitive research proposals (e.g., NSF CAREER program). Once the proposed ideas have been conceptualized, in Year 3–5, the PI will study security of hardware/software-based architectures such as real-time hypervisors [26, 27] and TrustZone-enabled RTS [28, 29]. In Year 5–7, the PI intends to study security/privacy issues of distributed cyber-physical systems such as UAV swarms [30] and vehicular networks [31]. From Year 7 onward, the PI intends to investigate security and resiliency aspects of more general-purpose systems along with IoT/edge-style cyber-physical computing platforms and study emerging technologies such as smart manufacturing, autonomous/electronic vehicles, and robot-aided automation (especially for elderly/disable people). **Note:** we present the plans beyond Year 2 to show the potentials of our proposed research agenda and the PI's long-term goals.

Strengthening Collaborative Efforts. The PI is developing new collaborations with other researchers for interdisciplinary work. In his first nine months as a faculty, the PI has already initiated collaborations with Dr. Sergio Salinas (Wichita State), Dr. Gedare Bloom (U. of Colorado at Colorado Springs), and Dr. Shubhra Kanti (Auburn) on topics related to security and privacy of cyber-physical systems and human-robot interaction. We anticipate to lead multiple joint NSF proposals in the future. These projects will not only solve complex, practical problems and contribute to the growth and development of computing research but also allow the PI to establish a successful academic career.

During his PhD, the PI has successfully collaborated with industrial research labs (SRI International and Toyota Motors) that result in multiple publications [31, 32] and a patent [33]. The PI intends to strengthen his ties with industrial research and contribute to improving security of consumer products. Hence, there is a high likelihood that the PI's future research outcomes will be deployed and improve the security and resiliency of critical cyber-physical systems.

Lack of Support for Seeding Future Research. The PI devotes his career to building secure, trustworthy, and resilient cyber-physical computing platforms. An integral part of the PI's career goals is to inspire, educate, and mentor the broader community including K-12, undergraduate, and graduate students, and foster equity and diversity in STEM education. The PI intends to promote reproducibility in systems research and will continue to disseminate his research findings through open-source initiatives. **The PI does not have sufficient funds[1] to support a doctoral student and initiate this research plans. The funding support from NSF for the proposed research initiation activities will be one of the significant steps towards achieving the PI's long-term career goals.**

1.3 PI's Expertise

and

The PI, with his diverse research background, is in a unique position to carry out the proposed research agendas. The PI's research expertise includes *(a)* real-time cyber-physical systems security [29, 32, 34–39], *(b)* development of resilient cyber-physical networks [40–42], and *(c)* resource management in cellular wireless networks [43–47]. In recent years, the PI has been at the forefront of the research in real-time security and resiliency with contributions ranging from developing theoretical models [34–37, 40–42] as well as designing system architectures [29, 38, 39]. His work on integrating security as first-class principle of real-time schedulers [34] has won the *outstanding paper* and *best student paper* awards at the IEEE RTSS. The PI has published his solutions in top real-time and networking conferences/journals including RTSS [34, 40], ECRTS [35], ICCPS [38], INFOCOM [42], and IoT [39]. The *PI has all the resources* (e.g., research computers, experimental platforms, and lab facilities) needed to complete this project successfully. The PI also has experience working with *underrepresented communities* in computing, most notably *women students* at all levels — doctoral, masters, undergraduate, and even middle school students. Further, the PI will leverage Wichita State's existing outreach programs (e.g., Shocker Engineering Academy [48] and Hub for Cybersecurity Education and Awareness [49]) as well as resources form other organizations (e.g., CRA's undergraduate mentoring programs [50] and NSA's K-12 initiatives [51]) and work on broadening participation of high school students and minority communities in engineering. Hence, the PI is ideal for developing real-time security techniques such as those proposed in this project and fostering diversity in STEM education.

Networked?
[?]
→ bedded

→ TWC?

2 Background and Related Work

Real-Time Systems. RTS are defined by their strong timing requirements. Some of the common properties and assumptions related to RTS are as follows: *(a)* implemented as a system of periodic tasks, *(b)* worst-case bounds are known for most loops as well as the critical pieces of code, *(c)* scheduled by priority-driven schemes, and *(d)* limited resources (e.g., processor, memory, energy). Each real-time task τ_i is characterized by (T_i, C_i, D_i) in which T_i is the period (inter-arrival time), C_i is the constant upper bound on the execution time, and D_i is the timing constraint (deadline). In multicore systems, task scheduling can be viewed as solving an allocation problem depending on design criteria such as [11]: *(a) no migration* (tasks are allocated to a fixed core), *(b) task-level migration* (the jobs of a task may execute on different cores), and *(c) job-level migration* (the jobs of a task migrate to and execute on different core although parallel execution of a job is not permitted). Schedulability tests [**?**, 11, 52, 53] are used to determine if all tasks in the system meet their respective deadlines. If this is the case, then the task set is deemed to be *"schedulable"* and the system, *safe*.

add SAE Paper

Related and Complementary Work. Enhancing security in real-time applications is an active research area (see the related surveys [54, 55]). There exists some work [56–67] on reconciling the addition of security mechanisms into RTS. However, they are designed for single core platforms only, and unlike ours, they require modification of the existing real-time tasks. Researchers also proposed architectural frameworks [22–24, 39, 68, 69] that use hardware/software mechanisms to protect against security vulnerabilities. However, those frameworks require custom hardware-support for monitoring, and hence

[1] The PI's startup funding supports only one graduate student for 24 months.

not suitable for legacy or off-the-shelf systems. Integrating monitoring and detection tasks for multicore RTS without custom hardware support is an open research problem. There also exist large number of work for generic cyber-physical/Internet-of-things-specific embedded systems (too many to enumerate here, refer to the related surveys [70–74]) — however the consideration of real-time security aspects distinguish proposed work from other research.

3 Integrating Monitoring and Detection

To begin with, we propose the following two performance criteria for integrating security tasks.

– ***Monitoring Frequency:*** Recall from our earlier discussion (Section 2) that application tasks in RTS are periodic (i.e., they follow a fixed inter-arrival pattern) and scheduled by a priority-driven order. In order to provide the best protection, the security tasks need to be executed quite often (i.e., smaller inter-arrival duration). On the one hand, if the interval between consecutive monitoring events is too large, the adversary may harm the system (and remain undetected) between two invocations of the security task. On the other hand, if the security tasks are executed very frequently then it may impact the schedulability of the real-time tasks. Herein lies an important *trade-off* between *monitoring frequency* and *schedulability*.

– ***Responsiveness:*** In some circumstances, a security task may need to execute with less interference (i.e., interruptions) from higher-priority tasks. For instance, consider the scenario where a security breach is suspected. In such an event the security task may be required to *perform more fine-grained checking instead of waiting for its next periodic slot*. This may result in delayed execution of low-priority, non-critical, real-time tasks. However, the scheduling policy needs to ensure that the system remains secure without violating real-time constraints for critical, high-priority, real-time tasks.

One may wonder why we cannot schedule the security tasks in the same way that the existing real-time tasks are scheduled. However, without careful schedulability analysis if we set arbitrary high-priority or small inter-arrival duration, this may violate timing constraints for the critical real-time tasks — thus the main safety requirements of the system will be threatened. [5] Unlike single core systems, integrating security into multicore platforms is more challenging since the designers now have multiple choices for where to allocate the security tasks. For instance, should the engineers: *(i) dedicate a core* to all the security tasks; or *(ii) spread the security tasks to all cores* (in conjunction with the real-time tasks) and if so, how to determine the *task-to-core assignment*? or *(iii) execute them continuously* across any available core? In addition, how the designers can *determine the monitoring frequency of security tasks*? How do we develop a *reactive framework that dynamically adjust security tasks into available cores for better responsiveness upon suspicion of a security breach?* These are themselves critical research challenges that need to be investigated.

\cite{ Section
-deploy }

Adversary Model. We assume that an adversary may destabilize the system by leveraging (known) vulnerabilities. Our focus is on threats that can be dealt with by *integrating additional security tasks* into the host (see Table 1 for related examples). The addition of such tasks may necessitate changing the schedule or increasing the execution time of real-time tasks as was the case in earlier work [56, 57, 60, 60, 62–64]. In this research we consider situations where additional security tasks are only allowed to have no (or minimal) impact on the schedule of existing real-time tasks and are not allowed to modify real-time parameters. We note that the design of integration techniques and the design of the specific security tasks are orthogonal problems; and our proposed techniques are agnostic to the specific monitoring mechanism.

Preliminary Work. The PI's dissertation work [34–37] provides the foundation of this research. In early work [34] we introduce a technique that allows the execution of security tasks *"opportunistically" with lowest-priority* in conjunction with real-time tasks (so that they do not interfere the execution order of real-time tasks), while keeping the best possible periods (for the security tasks. (i.e monitoring frequency)) However, if the security tasks always execute with lowest priority, they suffer more interference (i.e., preemption from high-priority real-time tasks) and the longer detection time (due to poor response time) will make the security mechanisms less effective. In order to provide better responsiveness and increase the effectiveness of monitoring and detection mechanisms, we then propose a *dual-mode framework* [35] that allows the security tasks to execute in two different modes: *(a)* by default security tasks execute opportunistically when the system is deemed to be uncompromised (PASSIVE mode); *(b)* if an anomaly is suspected, the security tasks may switch to higher priority (while ensuring the timing guarantees of real-time tasks) to perform additional checks as needed (ACTIVE mode); *(c)* the system reverts to PASSIVE mode if: *(i)* no anomalous activity is found or *(ii)* the root cause of the problem is detected and malicious entities are removed. We also devise a *metric* (called *"tightness of monitoring"*) that allows us to execute security tasks with a frequency closer to what the designers' expect and provides us one way to trade-off security with schedulability.

The above pieces work is designed for single core platforms. In the followup work [36] we first develop a low-complexity iterative solution that jointly finds the security tasks' periods and core assignments where the security tasks are *statically assigned* across all available cores. We then introduce with an alternate design mechanism [37] that *can raise the "responsiveness" of monitoring tasks by increasing their frequency of execution*. The key intuition is that if the security tasks are able to execute with as few interruptions as possible (e.g., by moving immediately to an empty core when they are interrupted), then there is much higher chance of successful detection and correspondingly, a much lower chance of successful adversarial action. The latter scheme provides better monitoring but comes with a cost (in terms of context switch overhead).

4 Proposed Work

4.1 Reactive Security Mechanisms for Multicore Systems

→ Introduced in our prior work

Recall that ~~existing~~ multicore security integration frameworks [36, 37] do not support runtime mode changes upon suspecting security breaches. Integrating reactive security mechanisms for legacy multicore platforms (where designers have less flexibility for changing system architecture/parameter) is still an open problem. For instance, will the security tasks *always be running* (perhaps on one of the dedicated cores)? Must the security tasks be allowed to *take up more (or all) cores* (e.g., switch to ACTIVE mode) as it detects malicious activity? We need to consider various design choices and implementation issues, such as:

(i) Should we take over one core complete to execute security tasks in ACTIVE mode (and perhaps reschedule the real-time tasks to other cores), or, use all the cores (or a subset of cores) for security tasks and perhaps execute with a higher priority than some of the real-time tasks? How quickly the intrusions could be detected?

↳ How do we determine these periods ?

(ii) Is it better to design a security task (that monitors continuously) that executes in a given core (for PASSIVE mode) or move across all cores, if so, does it improve schedulability/security? How the desired real-time requirements and control system performance can be satisfied with mode changes?

(iii) How the reactive security mechanisms (say PASSIVE-to-ACTIVE and vice-versa) will be implemented in practice in the RTOS/scheduler? What is the scheduling/context switch overhead of the mode switch?

To address these issues, we will *(a)* devise design-time analytical models to ensure that timing guarantees are met and *(b)* develop necessary APIs, libraries, and OS patches to deploy reactive security protocols into commodity real-time kernels. In particular, we will extend existing multi-mode real-time scheduling techniques [75–77] with security requirements and explore various design alternatives that mentioned above. * The development of our proposed reactive monitoring framework also requires system-level modifications. For instance, the scheduler must supports dynamic mode changes (with bounded latency) when it receives signal from corresponding security tasks. As part of our research, we will modify existing open-source RTOS schedulers (RT_PREEMPT Linux [78] and LITMUS^{RT} [79]) and provide support for continuous monitoring and dynamic mode changes.

* As mentioned earlier,

4.2 Atomicity and Task Dependency

Once we have developed the above reactive security mechanism for multicore RTS, we will extend our framework to *(a)* allow atomic (i.e., non-preemptive) operations and *(b)* support inter-dependent execution. Security tasks so far have been treated as being independent and preemptible. Depending on the operation, some security tasks may need to be executed *without preemption* (atomic operation). As an example, consider a security task that scans the process table and has been preempted in the middle of its operation. An adversary may corrupt the process table entry that has already been scanned before the next scheduling

We will formulate the period selection as a constrained optimization problem and explore existing tools such as geometric programming models to solve them.

point of the security task. When the security task is rescheduled, it will start scanning from its last known state and may not be able to detect the changes in a timely manner. One way to support such case is the following: when any security task needs to perform special atomic operations, the priority of the tasks can be increased to one that is strictly higher than all (or some) of the real-time tasks, depending on the requirements of the checking event. Such atomic operations (using priority inversions), however, may compromise the timing constraints of some (or all!) of the real-time tasks. Hence, schedulability analysis needs to consider this. Besides, the scheduling policy should identify which real-time or security tasks can be dropped (or perhaps reschedule to other cores) to provide better trade-off between real-time performance and security defense.

~~So~~ Add the example that commented out

Further, security tasks may have dependencies where one task depends on the output from one or more other tasks (i.e., they may need to follow certain *precedence constraints*). In such cases we may not independently execute the security tasks in parallel into multiple cores. To address this, we will use a directed acyclic graph (DAG) to capture the dependencies and constraints among security tasks. In this case, the "tightness" metric used in our earlier work [34–37] may no longer be a reasonable metric. We will include the constraints to ensure that the entire DAG is executed often and study what metrics can better capture security dependency requirements.

(see Section 3)

4.3 Security Metrics and Performance Evaluation

In preliminary work [34–37] we tested *time-to-detect an intrusion* as a performance metric to observe how well the security tasks can perform desired monitoring and detection after period adaptation. But how do we know that this trade-off framework works well in practice? How do we know what the efficient design choices (say core-to-task assign strategy)? What are the *right metrics* to measure the performance of the system? We need a way to measure the system performance and a methodical way of exploring the design space space. ~~Our goal is to explore the possible ways in which security could be integrated in multicore-based real-time platforms.~~ Apart from security metrics, it is also necessary to understand the impact of adding security on real-time timing properties to verify that the system functionality has not been impaired. For instance, albeit for a limited time, the dynamic mode switching may reduce utilization (especially for some low-priority real-time tasks in the ACTIVE mode). Hence, this *difference in utilization* can be used as a metric to measure the cost of security. While *time-to-detect* is a useful metric, it is hard to quantify in a comprehensive way as it depends on a number of factors (such as the efficacy of monitoring tasks, the kind of intrusion) and is a lagging metric. Identifying and designing better metrics is an important and challenging problem. We will undertake it in the narrow context of integrating monitoring and detection tasks into multicore RTS.

multicore

move to first task

4.4 Bringing it Together: Secure Hardware-Software Integration

All of the algorithms and mechanisms discussed thus far must be implemented as software components that run on the multicore platform. There is also requirement that security tasks are executed in tamper-resistance space such that an adversary cannot easily evade the ~~intrusion~~ detection mechanisms. If the security tasks are infiltrated or they cannot be scheduled, no matter how effective the original algorithms, we cannot guarantee the security of the system. One challenge is: *what are the minimum components that must be provided with the isolation mechanisms?* Even if the various components are carrying out their expected functionality, the security of the systems might be jeopardized if the underlying execution platforms (hardware and OS) are unable to protect security tasks from malicious actions. Hence, security tasks require *isolation guarantees* — to prevent tampering, and there is a need for trusted execution environments (TEEs) [80] to ensure the integrity and trustworthiness of the overall system. ~~In particular,~~ Hence, we propose to use off-the-shelf trusted modules (~~e.g.,~~ ARM TrustZone [73, 74]) to ensure tamper-proof execution of security tasks.[2] Since vanilla TrustZone is not designed for real-time applications, we need algorithms and techniques to ensure that security tasks comply with real-time requirements when they execute within the trusted enclaves. For instance, the scheduler and context switch overhead of

[2] Considering the short time-frame of this project, we limit our study on TrustZone-based enclaves. However, our ideas are rather general and can be adapted to other TEE architectures such as Intel SGX [81] and RISC-V MultiZone [82].

We will develop design-time Schedulability ~~analysis~~ models that analytically guarantee that the timing constraints are satisfy

rel

TrustZone enclaves ~~need to be~~ further bounded to ensure that timing constrains are not violated. In the proposed research we will develop *a secure OS-level interface* based on commodity open-source real-time kernels (RT_PREEMPT Linux [78] and LITMUS^{RT} [79]) and OP-TEE [83] TrustZone ports. Our scheduler plug-ins and OS-level mechanisms will ensure that security tasks are executed in a tamper-proof environment (TrustZone enclaves) and carried out their desired functionalities. In the past we *successfully used off-the-shelf trusted modules* in multiple cyber-physical use-cases [29,39] — we will leverage our expertise on TEEs in the proposed research. We will *open-source our implementation and release the APIs* for community use.

scheduler study,

while retaining real-time guarantees

Research Tasks

- **TASK 1.** Extend multi-mode framework for multicore RTS — this includes: *(a)* design-space exploration of multiple mode-switch alternatives, *(b)* integrate with RTOS and multicore schedulers, and *(c)* analyze the efficacy of each of the techniques.
- **TASK 2.** Design frameworks for methodically integrating reactive monitoring and detection mechanisms for multi-core RTS with non-preemptive execution and precedence constraints.
- **TASK 3.** *(a)* Explore different performance metrics and evaluate their efficacy; *(b)* Study whether new metrics can provide additional real-time/security insights.
- **TASK 4.** *(a)* Retrofit TrustZone technology for the real-time security integration frameworks — this includes development of new frameworks, algorithms, and schedulability analysis as well as necessary system-level modifications; *(b)* Develop OS-level APIs and integrate them with existing real-time kernels; *(c)* Prepare necessary documentation for open-source release.

5 Evaluation

We will evaluate our proposed schemes using a variety of methods as we present below. Our implementation details and source code will be *publicly available*.

Simulation-based Evaluation. All of our approaches will first be evaluated using an *internally developed simulator*. We will use the simulator to test with a more diverse set of real-time task parameters to explore the design-space and scalability of the solutions/algorithms that have been developed. In the past, we extensively use open-source and custom in-house simulators [34, 36, 40–42, 44–47, 84] primarily to test the theories, and we will use similar techniques to evaluate our proposed schemes.

System Integration and Benchmarks. We will integrate our proposed ideas into *two real-time kernels* (RT_PREEMPT [78] and LITMUS^{RT} [79]) and use OP-TEE [83] as our secure enclave. We will develop kernel patches and evaluate them on both x86 (UP Extreme [85]) and ARM (Raspberry Pi [86]) embedded hardware platforms. We will use existing intrusion detection tools such as Tripwire [15], AIDE [16], and Bro [17] as security tasks. Further, we will evaluate the efficacy of our scheduler implementations using *three embedded benchmark suites*: PapaBench [87], MiBench [88], and MultiBench [89]. These tools and benchmarks will enable us to evaluate the performance of models with realistic workloads. We will use *multiple performance indicators* (such as context switch and memory overheads, scheduling delays) to evaluate the trade-offs of integrating security into existing real-time schedulers.

Demonstrative Platforms. While our simulation-based studies focus on the scalability of the algorithms, we also aim to analyze the feasibility of our ideas in realistic environments. We will evaluate our ideas on *two off-the-shelf cyber-physical platforms* (see Figure 3): *(a)* a multi-terrain ground rover (MonsterBorg [90]) and *(b)* a six-degree-of-freedom robotic arm (ROT3U [91]). Our implementations will demonstrate the feasibility of security integration techniques on wide range of use-cases such as remote surveillance, home automation, and smart manufacturing.

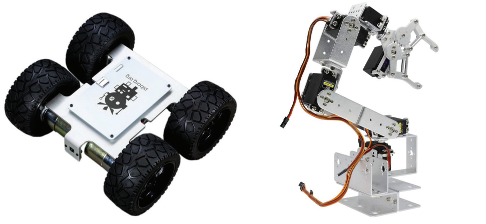

Figure 3: Off-the-shelf testbeds: *(a)* multi-terrain rover (left) and *(b)* six-degree-of-freedom robotic arm (right).

6 Project Execution Plan

Table 2 presents the timeline of this project. The PI has already started mentoring the PhD student who would be supported by this project. The PI and students will meet at least once a week. All students will congregate and jointly develop testbeds as well as work on the implementation. The PI will also participate in curriculum development and outreach activities (see Section 7) throughout the project. We will disseminate our research results from this work through scientific publications at *top real-time conferences* (RTSS, RTAS, ECRTS) and *multidisciplinary journals* (IEEE Access, IEEE TC), technical reports, open-source releases, prototype demonstrations/videos, and outreach activities.

Description	**Timeline**							
	Year 1				**Year 2**			
	Q1	**Q2**	**Q3**	**Q4**	**Q1**	**Q2**	**Q3**	**Q4**
TASK 1: Design multi-mode security integration framework	✓	✓						
TASK 2: Support atomiticity and task dependency			✓	✓				
TASK 3: Devise security metrics			✓	✓	✓	✓		
TASK 4: Develop TrustZone-enabled monitoring framework				✓	✓	✓	✓	
Testbed development and performance evaluation		✓	✓	✓	✓	✓	✓	✓
Education and outreach	✓	✓	✓	✓	✓	✓	✓	✓

Table 2: Timeline of our proposed research.

Plans for Assessing Success. We will evaluate our proposed research agendas based on the successful development of analytical models, performance metrics, and scheduler implementations. We will also track the number of citations of our publications, monitor visits to our project website, and observe the number of downloads/forks of our public code repositories. We will measure the success of our education plans by monitoring the number of student enrollment, course evaluations, ratings/comments/feedback from the students, and successful completion of course projects that turn into research publications. Our evaluation of outreach activities includes *(a)* monitoring the number of K-12 and undergraduate student participation and *(b)* successful development of new pedagogical contents. We will continuously monitor and evaluate our education and outreach activities and adjust them based on the observed results.

7 Broader Impacts

Societal Impact. The proposed research and educational plans have the potential for broad impact. A large number of critical systems of modern society have real-time requirements. They are also increasingly becoming targets for cyber attacks [3–10, 92]. Any successful, profound attacks on these systems can have catastrophic results, leading to loss of injury to humans, negative impacts on the system and even the environment. Hence, techniques developed as part of this project will make such systems *more secure* and applicable to a *broad range of domains* (e.g., avionics, automobiles, power grids, manufacturing plants, medical devices, industrial control systems, unmanned vehicles). Our proposed research can lead to substantial improvements in consumer products and also improve systems that have *national security considerations*. All data, insights, results, implemented frameworks, and course modules will be made available to the *research community via a dedicated project website* along with appropriate documentation. Our public releases will facilitate the rapid dissemination and adoption of our ideas in the scientific community. In addition to technical contributions, our proposed educational and outreach activities (presented next) will have a broad and positive societal impact by *(a)* training a qualified workforce and *(b)* encouraging underrepresented students to pursue careers in computing.

not italic

Curriculum Development. This proposal will feed into the PI's ongoing efforts to improve teaching real-time and cyber-physical systems, especially security concepts in these domains. Students supported by this project will get a solid scientific foundation in designing and implementing real-time security solutions. Other students (not directly involved in the research) will also benefit through research seminars, scientific publications, and class lectures. The PI anticipates that the results obtained from this project will provide a foundation for *one or more chapters in textbooks*.

◇ ***Development of a New Course:*** As an effort to expose students to diverse computing domains, the PI will develop a *new senior undergraduate/graduate-level course* on “Real-Time Embedded Systems”. The course will cover the basics of real-time scheduling and related security/safety principles as well as include hands-on lab exercises using popular RTOS (LITMUS^{RT} [79] and FreeRTOS [93]). The PI plans to design the course by Year 1 and teach Year 2 (Fall semester).

◇ ***Enhancement of Existing Courses:*** Introduction to Cybersecurity (CS 656) is one of the foundational courses in our undergraduate program and forms a prerequisite for many advanced graduate courses. The PI will extend the existing CS 656 syllabus and include *four lectures* on various attacks and related defense techniques on real-time and cyber-physical systems. Further, the PI will design a new *“red-team/blue-team” lab exercise* where red-team will develop techniques to break a real system (LITMUS^{RT} scheduler) and the blue-team will prevent this using the ideas being developed in this research. The PI intends to offer the revised CS 656 course from Year 2 (Spring semester). During the Spring 2021 semester, the PI introduced a graduate-level course on real-time security (CS 898CC) [94] that focused on recent research. As students from multiple areas were enrolled, the course was successful, and the class had very insightful discussions that generated multiple research ideas. The PI will offer the CS 898CC course again in Year 1 and Year 2 (Fall semesters), with additional topics on security integration techniques proposed in this project.

Undergraduate Research Involvement. This project will provide unique research opportunities for our undergraduate students. The PI will closely supervise the students — they will engage in semester-long research activities and assist the PhD student to *(a)* build the hardware prototypes, *(b)* deploy and test controller programs, and *(c)* prepare open-source releases and documentation. The PI will actively involve undergraduate students in the scientific writing process and include them as authors in the publications. They will also participate in Wichita State’s undergraduate research competition, URCAF [95], and present their work to the broader community.

Diversity and Outreach. The PI is attentive to diversity and inclusion in his research and educational activities. The PI currently advises *two female students* (one PhD and one undergraduate). The proposal includes *funding support for one female PhD student* who has *already started working on the project*. The PI will make additional efforts to promote diversity and ensure that *at least one position will be reserved all times in his research lab for underserved populations, including students with disabilities*.

◇ ***Broadening Participation in Wichita***[3]***:*** The PI is working closely with Wichita State’s Shocker Engineering Academy [48] to increase the representation of minority communities in engineering. The program is supported by an NSF initiative called KS-LSAMP (Kansas Louis Stokes Alliances for Minority Participation) [96]. As a part of this program, the PI is currently mentoring *two undergraduate students* from underrepresented communities (including *one black female* student). The PI will continue this effort and offer projects especially targeted for *freshman/sophomore undergraduate students* on topics related to security and resiliency of real-time and cyber-physical systems. The PI is also actively involved in Wichita State Hub for Cybersecurity Education and Awareness (HCEA) [49]. As a part of this initiative, PI will organize *CTF (capture the flag) competitions and security boot camps for middle and high school students* (Year 1 and Year 2, Summer semesters). The proposed initiatives will develop research mindset for K-12 and undergraduate students from the very early stages of their academic careers.

◇ ***Other Outreach Activities:*** The PI will *mentor underrepresented students* by partnering with the CRA’s Distributed Research Experiences for Undergraduates (DREU) program [50]. The PI will apply to be a DREU faculty mentor at the beginning of the project and intends to mentor the students in both years. The PI will also *volunteer for presentations and demos* at events organized by government agencies such as NSA’s CAE K-12 CyberTalk [51] throughout the project to involve minority groups and high school seniors.

8 Results from Prior NSF Support

The PI has not received any grant or contract from the federal government.

[3] Letters from *Ana Lazarín* (Director, Broadening Participation and Recruitment, College of Engineering) and *Dr. Joe Jabara* (Director, Hub for Cybersecurity Education and Awareness) are attached.

Check Refs, Especially MDPI

REFERENCES CITED

[1] A. Hussain, M. Hannan, A. Mohamed, H. Sanusi, and A. Ariffin, "Vehicle crash analysis for airbag deployment decision," *Int. J. of Auto. Tech.*, vol. 7, no. 2, pp. 179–185, 2006.

[2] K. Castelli, A. M. A. Zaki, and H. Giberti, "Development of a practical tool for designing multi-robot systems in pick-and-place applications," *MDPI Robotics*, vol. 8, no. 3, 2019.

[3] N. Falliere, L. O. Murchu, and E. Chien, "W32. stuxnet dossier," *White paper, Symantec Corp., Security Response*, vol. 5, p. 6, 2011.

[4] R. M. Lee, M. J. Assante, and T. Conway, "Analysis of the cyber attack on the ukrainian power grid," *SANS Industrial Control Systems*, 2016.

[5] K. Koscher, A. Czeskis, F. Roesner, S. Patel, T. Kohno, S. Checkoway, D. McCoy, B. Kantor, D. Anderson, H. Shacham *et al.*, "Experimental security analysis of a modern automobile," in *IEEE S&P*, 2010, pp. 447–462.

[6] S. Checkoway, D. McCoy, B. Kantor, D. Anderson, H. Shacham, S. Savage, K. Koscher, A. Czeskis, F. Roesner, T. Kohno *et al.*, "Comprehensive experimental analyses of automotive attack surfaces," in *USENIX Sec. Symp.*, 2011.

[7] H. Teso, "Aircraft hacking: Practical aero series," in *HITB Sec. Conf.*, 2013.

[8] D. P. Shepard, J. A. Bhatti, T. E. Humphreys, and A. A. Fansler, "Evaluation of smart grid and civilian UAV vulnerability to GPS spoofing attacks," in *Proc. of the ION GNSS Meeting*, vol. 3, 2012.

[9] J. Valente and A. A. Cardenas, "Understanding security threats in consumer drones through the lens of the discovery quadcopter family," in *ACM IoTS&P*, 2017, pp. 31–36.

[10] S. S. Clark and K. Fu, "Recent results in computer security for medical devices," in *MobiHealth*, 2011, pp. 111–118.

[11] R. I. Davis and A. Burns, "A survey of hard real-time scheduling for multiprocessor systems," *ACM CSUR*, vol. 43, no. 4, pp. 35:1–35:44, 2011.

[12] L. Sha, M. Caccamo, R. Mancuso, J.-E. Kim, M.-K. Yoon, R. Pellizzoni, H. Yun, R. B. Kegley, D. R. Perlman, G. Arundale *et al.*, "Real-time computing on multicore processors," *IEEE Comp.*, vol. 49, no. 9, pp. 69–77, 2016.

TC?

[13] O. Kotaba, J. Nowotsch, M. Paulitsch, S. M. Petters, and H. Theiling, "Multicore in real-time systems–temporal isolation challenges due to shared resources," in *DATE*, 2013.

[14] P. K. Valsan, H. Yun, and F. Farshchi, "Addressing isolation challenges of non-blocking caches for multicore real-time systems," *Springer RTS*, vol. 53, no. 5, pp. 673–708, 2017.

[15] "Tripwire," https://github.com/Tripwire/tripwire-open-source.

[16] "AIDE," http://aide.sourceforge.net/.

[17] "The Bro network security monitor," https://www.bro.org.

[18] M. Roesch, "Snort - lightweight intrusion detection for networks," in *USENIX Conf. on Sys. Admin.*, 1999, pp. 229–238.

[19] V. M. Weaver, "Linux perf_event features and overhead," in *IEEE FastPath*, 2013.

[20] "OProfile," http://oprofile.sourceforge.net/.

[21] V. Chandola, A. Banerjee, and V. Kumar, "Anomaly detection: A survey," *ACM CSUR*, vol. 41, no. 3, p. 15, 2009.

[22] M.-K. Yoon, S. Mohan, J. Choi, J.-E. Kim, and L. Sha, "SecureCore: A multicore-based intrusion detection architecture for real-time embedded systems," in *IEEE RTAS*, 2013, pp. 21–32.

[23] M.-K. Yoon, S. Mohan, J. Choi, and L. Sha, "Memory heat map: anomaly detection in real-time embedded systems using memory behavior," in *ACM/EDAC/IEEE DAC*, 2015, pp. 1–6.

[24] M.-K. Yoon, S. Mohan, J. Choi, M. Christodorescu, and L. Sha, "Learning execution contexts from system call distribution for anomaly detection in smart embedded system," in *ACM/IEEE IoTDI*, 2017, pp. 191–196.

[25] J. Song, G. Fry, C. Wu, and G. Parmer, "CAML: machine learning-based predictable, system-level anomaly detection," in *IEEE CERTS*, vol. 29, 2016.

[26] S. Pinto, J. Pereira, T. Gomes, A. Tavares, and J. Cabral, "LTZVisor: TrustZone is the key," in *Euromicro ECRTS*, 2017, pp. 4:1–4:22.

[27] S. Xi, J. Wilson, C. Lu, and C. Gill, "RT-Xen: Towards real-time hypervisor scheduling in Xen," in *ACM EMSOFT*, 2011, pp. 39–48.

[28] A. Mukherjee, T. Mishra, T. Chantem, N. Fisher, and R. Gerdes, "Optimized trusted execution for hard real-time applications on cots processors," in *ACM RTNS*, 2019, pp. 50–60.

[29] M. Hasan and S. Mohan, "Protecting actuators in safety critical IoT systems from control spoofing attacks," in *ACM IoT S&P*, 2019, pp. 8–14.

[30] G. Chmaj and H. Selvaraj, "Distributed processing applications for uav/drones: a survey," in *Springer Prog. in Sys. Eng.*, 2015, pp. 449–454.

[31] M. Hasan, S. Mohan, T. Shimizu, and H. Lu, "Securing vehicle-to-everything (V2X) communication platforms," *IEEE TIV*, vol. 5, no. 4, pp. 693–713, 2020.

[32] I. Agadakos, C.-Y. Chen, M. Campanelli, P. Anantharaman, M. Hasan, B. Copos, T. Lepoint, M. Locasto, G. F. Ciocarlie, and U. Lindqvist, "Jumping the air gap: Modeling cyber-physical attack paths in the internet-of-things," in *ACM CPS-SPC*, 2017, p. 37–48.

[33] G. F. Ciocarlie, I. Agadakos, C.-Y. Chen, M. Campanelli, P. Anantharaman, M. Hasan, U. Lindqvist, M. Locasto, B. Copos, T. Lepoint *et al.*, "Modeling cyber-physical attack paths in the Internet-of-things," May 21 2020, uS Patent App. 16/634,591.

[34] M. Hasan, S. Mohan, R. B. Bobba, and R. Pellizzoni, "Exploring opportunistic execution for integrating security into legacy hard real-time systems," in *IEEE RTSS*, 2016, pp. 123–134.

[35] M. Hasan, S. Mohan, R. Pellizzoni, and R. B. Bobba, "Contego: An adaptive framework for integrating security tasks in real-time systems," in *Euromicro ECRTS*, 2017, pp. 23:1–23:22.

[36] M. Hasan, S. Mohan, R. Pellizzoni, and R. B. Bobba, "A design-space exploration for allocating security tasks in multicore real-time systems," in *DATE*, 2018, pp. 225–230.

[37] M. Hasan, S. Mohan, R. Pellizzoni, and R. B. Bobba, "Period adaptation for continuous security monitoring in multicore systems," in *DATE*, 2020, pp. 430–435.

[38] F. Abdi, C.-Y. Chen, M. Hasan, S. Liu, S. Mohan, and M. Caccamo, "Guaranteed physical security with restart-based design for cyber-physical systems," in *ACM/IEEE ICCPS*, 2018, pp. 10–21.

[39] F. Abdi, C.-Y. Chen, M. Hasan, S. Liu, S. Mohan, and M. Caccamo, "Preserving physical safety under cyber attacks," *IEEE IoT J.*, vol. 6, no. 4, pp. 6285–6300, 2018.

[40] R. Kumar, M. Hasan, S. Padhy, K. Evchenko, L. Piramanayagam, S. Mohan, and R. B. Bobba, "End-to-end network delay guarantees for real-time systems using SDN," in *IEEE RTSS*, 2017, pp. 231–242.

[41] A. Kashinath, M. Hasan, S. Mohan, R. Bobba, and R. Mittal, "Improving dependability via deadline guarantees in commodity real-time networks," in *IEEE SecSDN*, 2020, pp. 1–6.

[42] A. Kashinath, M. Hasan, R. Kumar, S. Mohan, R. B. Bobba, and S. Padhy, "Safety critical networks using commodity SDNs," in *IEEE INFOCOM*, 2021.

[43] M. Hasan and E. Hossain, "Distributed resource allocation in 5G cellular networks," in *Towards 5G: Applications, Requirements and Candidate Technologies*, 1st ed., R. Vannithamby and S. Talwar, Eds. Wiley, 2017, ch. 8, pp. 129–161.

[44] M. Hasan and E. Hossain, "Distributed resource allocation for relay-aided device-to-device communication under channel uncertainties: A stable matching approach," *IEEE TCOM*, vol. 63, no. 10, pp. 3882–3897, 2015.

[45] M. Rasti, M. Hasan, L. B. Le, and E. Hossain, "Distributed uplink power control for multi-cell cognitive radio networks," *IEEE TCOM*, vol. 63, no. 3, pp. 628–642, 2015.

[46] M. Hasan and E. Hossain, "Distributed resource allocation for relay-aided device-to-device communication: A message passing approach," *IEEE TWC*, vol. 13, no. 11, pp. 6326–6341, 2014.

[47] M. Hasan, E. Hossain, and D. I. Kim, "Resource allocation under channel uncertainties for relay-aided device-to-device communication underlaying LTE-A cellular networks," *IEEE TWC*, vol. 13, no. 4, pp. 2322–2338, 2014.

[48] "Wichita State University Shocker Engineering Academy (SEA)," https://www.wichita.edu/academics/engineering/sea/index.php.

[49] "Wichita State Hub for Cybersecurity Education and Awareness (HCEA)," https://www.wichita.edu/academics/engineering/cybersecurity/index.php.

[50] "Distributed Research Experiences for Undergraduates (DREU)," https://cra.org/cra-wp/dreu/.

[51] "NSA CAE K12 CyberTalk," https://k12cybertalk.org.

[52] N. Audsley, A. Burns, M. Richardson, K. Tindell, and A. J. Wellings, "Applying new scheduling theory to static priority pre-emptive scheduling," *SE Journal*, vol. 8, no. 5, pp. 284–292, 1993.

[53] E. Bini and G. C. Buttazzo, "Schedulability analysis of periodic fixed priority systems," *IEEE TC*, vol. 53, no. 11, pp. 1462–1473, 2004.

[54] C.-Y. Chen, M. Hasan, and S. Mohan, "Securing real-time Internet-of-things," *Sensors*, vol. 18, no. 12, 2018.

[55] H. Chai, G. Zhang, J. Zhou, J. Sun, L. Huang, and T. Wang, "A short review of security-aware techniques in real-time embedded systems," *J. of Cir., Sys. and Comp.*, vol. 28, no. 02, 2019.

[56] T. Xie and X. Qin, "Improving security for periodic tasks in embedded systems through scheduling," *ACM TECS*, vol. 6, no. 3, p. 20, 2007.

[57] M. Lin, L. Xu, L. T. Yang, X. Qin, N. Zheng, Z. Wu, and M. Qiu, "Static security optimization for real-time systems," *IEEE Trans. on Indust. Info.*, vol. 5, no. 1, pp. 22–37, 2009.

[58] V. Lesi, I. Jovanov, and M. Pajic, “Network scheduling for secure cyber-physical systems,” in *IEEE RTSS*, 2017, pp. 45–55.

[59] V. Lesi, I. Jovanov, and M. Pajic, “Security-aware scheduling of embedded control tasks,” *ACM TECS*, vol. 16, pp. 188:1–188:21, 2017.

[60] S. Mohan, M.-K. Yoon, R. Pellizzoni, and R. B. Bobba, “Real-time systems security through scheduler constraints,” in *Euromicro ECRTS*, 2014, pp. 129–140.

[61] R. Pellizzoni, N. Paryab, M.-K. Yoon, S. Bak, S. Mohan, and R. B. Bobba, “A generalized model for preventing information leakage in hard real-time systems,” in *IEEE RTAS*, 2015, pp. 271–282.

[62] S. Mohan, M.-K. Yoon, R. Pellizzoni, and R. B. Bobba, “Integrating security constraints into fixed priority real-time schedulers,” *Springer RTS*, vol. 52, no. 5, pp. 644–674, 2016.

[63] X. Zhang, J. Zhan, W. Jiang, Y. Ma, and K. Jiang, “Design optimization of security-sensitive mixed-criticality real-time embedded systems,” in *IEEE ReTiMiCS*, 2013.

[64] K. Jiang, P. Eles, and Z. Peng, “Optimization of secure embedded systems with dynamic task sets,” in *DATE*, 2013, pp. 1765–1770.

[65] M.-K. Yoon, S. Mohan, C.-Y. Chen, and L. Sha, “TaskShuffler: A schedule randomization protocol for obfuscation against timing inference attacks in real-time systems,” in *IEEE RTAS*, 2016, pp. 1–12.

[66] C.-Y. Chen, M. Hasan, A. Ghassami, S. Mohan, and N. Kiyavash, “REORDER: Securing dynamic-priority real-time systems using schedule obfuscation,” Tech. Rep., 2019. [Online]. Available: https://arxiv.org/pdf/1806.01393.pdf

[67] J. Chen, T. Kloda, A. Bansal, R. Tabish, C.-Y. Chen, B. Liu, S. Mohan, M. Caccamo, and L. Sha, “SchedGuard: Protecting against schedule leaks using linux containers,” in *IEEE RTAS*, 2021, pp. 14–26.

[68] S. Mohan, S. Bak, E. Betti, H. Yun, L. Sha, and M. Caccamo, “S3A: Secure system simplex architecture for enhanced security and robustness of cyber-physical systems,” in *ACM HiCoNS*. ACM, 2013, pp. 65–74.

[69] D. Lo, M. Ismail, T. Chen, and G. E. Suh, “Slack-aware opportunistic monitoring for real-time systems,” in *IEEE RTAS*, 2014, pp. 203–214.

[70] A. Humayed, J. Lin, F. Li, and B. Luo, “Cyber-physical systems security – A survey,” *IEEE IoT J.*, vol. 4, no. 6, pp. 1802–1831, 2017.

[71] Y. Yang, L. Wu, G. Yin, L. Li, and H. Zhao, “A survey on security and privacy issues in Internet-of-Things,” *IEEE IoT J.*, vol. 4, no. 5, pp. 1250–1258, 2017.

[72] M. Ammar, G. Russello, and B. Crispo, “Internet of Things: A survey on the security of IoT frameworks,” *Elsevier J. of Inf. Sec. & App.*, vol. 38, pp. 8–27, 2018.

[73] S. Pinto and N. Santos, “Demystifying ARM TrustZone: A comprehensive survey,” *ACM CSUR*, vol. 51, no. 6, p. 130, 2019.

[74] W. Li, H. Chen, and H. Chen, “Research on ARM TrustZone,” *ACM GetMobile*, vol. 22, no. 3, pp. 17–22, 2019.

[75] V. Nelis, J. Goossens, and B. Andersson, “Two protocols for scheduling multi-mode real-time systems upon identical multiprocessor platforms,” in *EUROMICRO ECRTS*, 2009, pp. 151–160.

[76] P. M. Yomsi, V. Nelis, and J. Goossens, "Scheduling multi-mode real-time systems upon uniform multiprocessor platforms," in *IEEE ETFA*, 2010, pp. 1–8.

[77] H. Baek, K. G. Shin, and J. Lee, "Response-time analysis for multi-mode tasks in real-time multiprocessor systems," *IEEE Access*, vol. 8, pp. 86 111–86 129, 2020.

[78] L. Fu and R. Schwebel, "Real-time Linux wiki," https://rt.wiki.kernel.org/index.php/rt_preempt_howto, [Online].

[79] J. M. Calandrino, H. Leontyev, A. Block, U. C. Devi, and J. H. Anderson, "LITMUS^{RT}: A testbed for empirically comparing real-time multiprocessor schedulers," in *IEEE RTSS*, 2006, pp. 111–126.

[80] M. Sabt, M. Achemlal, and A. Bouabdallah, "Trusted execution environment: What it is, and what it is not," in *IEEE Trustcom/ BigDataSE/ISPA*, 2015, pp. 57–64.

[81] V. Costan and S. Devadas, "Intel SGX Explained," *IACR Crypt. ePrint Arch.*, no. 086, pp. 1–118, 2016.

[82] C. Garlati and S. Pinto, "Secure iot firmware for RISC-V processors," in *Emb. Wor. Conf.*, 2022, pp. 1–4.

[83] "Open Portable Trusted Execution Environment," https://www.op-tee.org/.

[84] M. Hasan, S. Mohan, R. B. Bobba, and R. Pellizzoni, "A server model to integrate security tasks into fixed-priority real-time systems," in *IEEE CERTS*, 2016, pp. 61–68.

[85] "UP Xtreme Series," https://up-shop.org/up-xtreme-series.html/.

[86] "Raspberry Pi," https://www.raspberrypi.org/products/raspberry-pi-4-model-b/.

[87] F. Nemer, H. Cassé, P. Sainrat, J.-P. Bahsoun, and M. De Michiel, "Papabench: a free real-time benchmark," in *EUROMICRO WCET*, 2006.

[88] M. R. Guthaus, J. S. Ringenberg, D. Ernst, T. M. Austin, T. Mudge, and R. B. Brown, "Mibench: A free, commercially representative embedded benchmark suite," in *IEEE WWC-4*, 2001, pp. 3–14.

[89] "EEMBC MultiBench Multicore Benchmark Suite," https://www.eembc.org/multibench/.

[90] "MonsterBorg Raspberry Pi robot," https://www.piborg.org/robots-1/monsterborg.

[91] "ROT3U 6DOF aluminium robot arm," https://github.com/stawo/robot-arm.

[92] J. Westling, "Future of the Internet of things in mission critical applications," 2016.

[93] "FreeRTOS real-time operating system," https://freertos.org.

[94] "Wichita State University graduate course – CS898CC: Security for Real-Time Internet-of-Things," https://tinyurl.com/cs898ccsp21.

[95] "Wichita State University Undergraduate Research and Creative Activity Forum (URCAF)," https://www.wichita.edu/academics/research/urcaf/.

[96] "The National Science Foundation (NSF) Kansas Louis Stokes Alliance for Minority Participation (KS-LSAMP)," https://www.wichita.edu/about/conferences/LSAMP.php.

Budget justification.

FACILITIES, EQUIPMENT, AND OTHER RESOURCES

FACILITIES

The PI is a faculty member in the School of Computing (SoC). The proposed project leverages several computational infrastructures already in place at Wichita State University (WSU). The PI will utilize existing facilities, equipment, and resources to execute the proposed research successfully.

Cyber-Physical Systems Security Research Lab. The PI directs the Cyber-Physical Systems Security Research Lab (CPS2RL).[1] This 800 square feet lab is located on WSU's main campus in Wallace Hall, Room 331. The proposed research activities will be conducted in this lab. The lab contains various evaluation platforms, including multiple embedded development boards (ZedBoard, Raspberry Pi, BeagleBone Black, and UP Extreme) ~~running Linux operating systems~~, one FischerTechnik manufacturing testbed, and four 3D printers. The lab also includes four general-purpose workstations (2.4 GHz CPU, 8 GB RAM, and 256 GB solid-state hard drive) and necessary research spaces for graduate and undergraduate students. The students will use these development boards and workstations for simulations, prototype development, and experiments. The lab is available 24/7 for the PI and students. running Windows OS

talk about new boards that we order

Office Spaces and Administrative Supports. The PI has 100 square feet of private office space on WSU's main campus in Jabara Hall, Room 244. The office is appropriate for day-to-day activities and holding meetings for the project. The graduate and undergraduate students will have access to student cubicles in the PI's research lab (CPS2RL). The graduate students will also have access to student cubicles in Wallace Hall, Room 309. These offices are appropriate for the students' day-to-day scholarly activities required for the project. The PI and students also have secretarial and technical support services provided through their department.

Outreach Activities. The PI will leverage WSU's existing outreach programs (e.g., Shocker Engineering Academy and Security Education Hub) and work on broadening participation in computing (refer to the attached ~~support~~ letters). These resources are not included in the budget.

MAJOR EQUIPMENT

Not applicable for this project.

OTHER RESOURCES

Startup Support. The PI has $50,000 startup support for purchasing research equipment. The PI will use $5500 from this startup funds to purchase additional hardware required for this project: *(a)* ground rover and robotic arm testbeds ($3000) and *(b)* a high-end desktop computer for the PhD student ($2500).

General Resources. The Office of R~~r~~esearch at WSU promotes the expertise of the faculty by facilitating all aspects of externally funded grants and contracts. They will provide oversight, assistance, and guidance to administer this project once awarded.

Other Computing/IT Resources. WSU's high-performance computing cluster (BeoShock) has two large GPUs and 800 CPU cores. The cluster is available to the PI and students for large-scale simulations. All IT computing facilities are connected to the gigabit campus backbone. The PI and students have access to the computing facilities via Ethernet/Wireless connections. WSU also provides web spaces and dedicated URLs for hosting webpages. The university also provides video conferencing software (Zoom and Teams) that can be used to hold virtual meetings.

[1] https://cps2rl.github.io.

DATA MANAGEMENT PLAN

1 Types of Data

We will not use any data from other sources except those generated from our research activities. We expect the following data to be generated as a direct result of the work being proposed in this project:

- Algorithms, design documents, and code for proposed schemes.
- Metrics to analyze the effectiveness of security integration techniques and the evaluated values based on these metrics.
- Use cases and test harnesses for our implementations.
- Source code and documentation related to integrating security techniques in commodity operating systems (RT_PREEMPT, LITMUS^{RT}, and OP-TEE).
- Analyses and evaluation tools such as simulation frameworks and the design of hardware testbeds (ground rover and robotic arm).
- Design of experiments, experimental results, analysis, and videos for demonstrations.
- Reference designs for URCAF competition developed by undergraduate students.
- Student reports, project designs, software artifacts, and theses.
- Scientific publications with research findings.
- Course materials (e.g., lecture notes, presentation slides, exams, and project assignments).

2 Data and Metadata Standards

All data will be electronic. We will utilize industry standards and publicly accepted data formats. The descriptions, algorithms, theories, results will be documented in conference and journal papers, technical reports, research notes, and online files. Source files will be stored in appropriate formats for the specific applications. We will document source code in standard ways that indicate distribution license as well as source information. Data sets will be retained in their original form (binary, CSV, XML, JSON). Videos for demonstrations will be recorded in standard format (MP4). If the formats are not standardized, we will provide the relevant documentation of the data formats.

3 Policies for Access and Sharing and Provisions for Appropriate Protection/Privacy

The PI retains the right to use all data generated as part of this research and anticipates sharing of data through publications and open source releases. There will be no copyright or licensing issues associated with the data being collected and maintained. This study will only collect non-sensitive data. We will not collect any private data in this research, and no personal identifiers will be recorded or retained by the PI or ? students. Hence, there will be no privacy concerns.

Any vulnerabilities in existing systems, if discovered, will be dealt with in the following manner: *(a)* we will actively work with the vendors to develop patches/safeguards that can mitigate the issues; *(b)* we will further provide 3–6 months of lead times to vendors in case they wish to work on the fix themselves. The PI will only publish the results once these processes have been completed or lead times have lapsed.

4 Policies and Provisions for Re-use, Re-distribution

We will not enforce any permissions or restrictions on the released code and data as we hope to foster the open access initiatives. We will use popular open-source licenses such as Apache, BSD, GPL, and MIT for our released code. We will choose the specific license on a case-by-case basis depending on whether other open-source projects are used in the development.

5 Plans for Archiving and Preservation of Access

We will store initial code and data on computers/testbed machines where they are generated. We will periodically back up the data using versioning infrastructure (Git). Data will be archived on local servers/versioning websites (GitHub) and managed for long-term storage. We will preserve the code and data and open access to them for at least five years after the award. After five years, we will continue to preserve the code and data through regular backups and maintenance on our lab's computers/local storage servers as well as on a public archiving website (GitHub).

6 Policies for Dissemination and Sharing of Research Results

The PI is attentive to the broad dissemination of his research findings. By preserving and making project data accessible, we will inform the broader community and foster discovery and collaboration. Several source code and findings from the PI's prior research efforts are publicly available on GitHub (`https://github.com/mnwrhsn`). The PI currently maintains a public GitHub repository for his research group (`https://github.com/CPS2RL`). We will also create a dedicated website for this project under our departmental domain (`https://www.wichita.edu/academics/engineering/eecs/cps2rl/realtimesecurity`). We will provide contact information on the website to communicate with the PI directly. All data obtained from the research will be fully documented and publicly available on our project website and GitHub repositories. The documentation will describe our experimental settings, including how the experiments were conducted and how the data was measured. Our open-source releases will enable replication of the experiments by other researchers and practitioners.

SOC ?

The PI and graduate students will internally participate in research seminars at the university to highlight the latest/preliminary results. We will broadly disseminate significant research findings through scholarly publications. For instance, algorithms, research methods, implementation details, and results will be disseminated via conference, journal, and workshop publications. Electronic manuscripts of publications will be available for public access on our project website. We will provide links to the appropriate publisher in case copyright issues prevent us from uploading the publications. Extended research results will be made available via technical reports on our project website and archived on our institutional repository (`https://soar.wichita.edu`). Videos recorded for demonstrations will be made available via public streaming sites such as YouTube, and access links will be provided on the project website. Course modules and lecture slides developed based on this work will be publicly available on our website. We will not release instructor resources (such as solutions to lab problems and homework assignments) on our website; however, they will be made available to educators upon request.

All materials will contain an acknowledgment of NSF support as follows: *"This material is based upon work supported by the National Science Foundation under Grant No. (NSF grant number)."* Additionally, we will include a disclaimer that *"any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation."* The PI will also work with the Wichita State University's Office of Research to ensure that our activities conform to NSF's research dissemination and sharing policies/requirements.

POSTDOCTORAL MENTORING PLAN

Not applicable for this proposal.

COLLABORATORS AND OTHER AFFILIATIONS INFORMATION

Not applicable for this proposal.

PROJECT PERSONNEL AND PARTNER ORGANIZATIONS

- Monowar Hasan; Wichita State University; PI