INVITED PAPER

SURE: A Modeling and Simulation Integration Platform for Evaluation of SecUre and REsilient Cyber–Physical Systems

This paper describes a modeling and simulation environment that can be used to evaluate attacker–defender behavior.

By Xenofon Koutsoukos, *Senior Member IEEE*, Gabor Karsai, *Senior Member IEEE*, Aron Laszka, Himanshu Neema, Bradley Potteiger, Peter Volgyesi, Yevgeniy Vorobeychik, *Member IEEE*, and Janos Sztipanovits, *Life Fellow IEEE*

ABSTRACT | The exponential growth of information and communication technologies have caused a profound shift in the way humans engineer systems leading to the emergence of closed-loop systems involving strong integration and coordination of physical and cyber components, often referred to as cyber–physical systems (CPSs). Because of these disruptive changes, physical systems can now be attacked through cyberspace and cyberspace can be attacked through physical means. The paper considers security and resilience as system properties emerging from the intersection of system dynamics and the computing architecture. A modeling and simulation integration platform for experimentation and evaluation of resilient CPSs is presented using smart transportation systems as the application domain. Evaluation of resilience is based on attacker–defender games using simulations of sufficient fidelity. The platform integrates 1) realistic models of cyber and physical components and their interactions; 2) cyber attack models that focus on the impact of attacks to CPS behavior and operation; and 3) operational scenarios that can be used for evaluation of cybersecurity risks. Three case studies are presented to demonstrate the advantages of the platform: 1) vulnerability analysis of transportation networks to traffic signal tampering; 2) resilient sensor selection for forecasting traffic flow; and 3) resilient traffic signal control in the presence of denial-of-service attacks.

KEYWORDS | Cyber–physical systems (CPSs); modeling and simulation; security and resilience; transportation networks

I. INTRODUCTION

The exponential growth of information and communication technologies over the last decade has given rise to their expansion in real-world computing applications involving physical processes. This expansion has led to the emergence of closed-loop systems involving strong integration and coordination of physical and cyber components, often referred to as cyber–physical systems (CPSs) [1]. These systems are rapidly finding their way into various sectors of the economy, such as transportation, industrial control systems, healthcare, and critical infrastructure. Increasing dependence on CPS renders them critical, and in-turn demands them to be secure, robust, reliable, and trustworthy, but it also makes them very attractive targets for cyber attacks.

Manuscript received May 8, 2017; revised July 17, 2017; accepted July 17, 2017. Date of publication August 15, 2017; date of current version December 20, 2017. This work was supported by the Air Force Research Laboratory under Award FA 8750-14-2-0180 and by the National Science Foundation under Grant CNS-1238959. 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 Air Force Research Laboratory or the National Science Foundation. *(Corresponding author: Xenofon Koutsoukos.)*

The authors are with the Institute for Software Integrated Systems, Vanderbilt University, Nashville, TN 37212 USA (e-mail: xenofon.koutsoukos@vanderbilt.edu; gabor.karsai@vanderbilt.edu; aron.laszka@vanderbilt.edu; himanshu.neema@vanderbilt.edu; peter.volgyesi@vanderbilt.edu; yevgeniy.vorobeychik@vanderbilt.edu; janos.sztipanovits@vanderbilt.edu).

Digital Object Identifier: 10.1109/JPROC.2017.2731741

While CPS research addresses the tight interaction between the physical and cyber parts of from a performance point of view [2], in-depth consideration of security and resilience is a significant challenge. A multivector attack exploiting a combined set of vulnerabilities from individual components, none of which might pose a serious threat to the standalone component, can have damaging effects in the overall system. Much of the cybersecurity related studies and efforts have focused on the foundations and technology required for network and information security. However, the full scope of the required research in CPSs is much wider and deeper than a restructuring focusing on the cyber side. There is a profound revolution driven by technology and market forces that turns whole industrial sectors into producers of CPS. This is not about adding computing and communication equipment to conventional products where both sides maintain separate identity. This is about merging computing and networking with physical systems to create new capabilities and product qualities. Whether we recognize it or not, we are in the midst of a pervasive, profound shift in the way humans engineer physical systems and manage their physical environment using networking and information technology. Because of these disruptive changes, physical systems can now be attacked through cyberspace and cyberspace can be attacked through physical means.

To date, security and resilience have been considered as largely disjoint (frequently even totally missing) aspects of CPS design. This separation was natural due to the traditionally segmented nature of design flows along isolated aspects of physical and cyber (software and computing) design. However, modern CPSs do not permit such separation anymore due to advances and integration in wireless sensor–actuator networks, the internet of "everything," data-driven analytics, and machine-to-machine interfaces. These developments have given CPSs the ability to interoperate and adapt to open dynamic environments, and enabled new trends: 1) faster operational time scales; 2) greater spatial interconnectedness; 3) larger number of mixed initiative interactions; and 4) increased heterogeneity of components. These trends are forcing increasingly physical and cyber sides of systems to be tightly coupled. The failure of loosely coupled physical and cyber schemes is evident in chronically unresolved design conflicts between performance and resilience against faults and intrusions, and conflicts between needs for performance optimization while maintaining robustness against adversarial impacts.

Building on the remarkable progress achieved during the past decade in developing a new system science for CPS, the objectives of our work are to analyze the cybersecurity risks, propose resilient monitoring and control mechanisms, and evaluate their effectiveness as well as their performance impact on system operations. We consider security and resilience as system properties emerging from the intersection of system dynamics and the computing architecture. This integrative view allows pursuing cross-domain tradeoffs and system-security codesign. Resilient dynamics generalize functional performance by augmenting design concerns to attain robustness against faults and cyber attacks. The effects of failures and intrusions are modeled as uncertainties and cast as adversarial games. We investigate how to efficiently solve these games and design efficient defense strategies against worst case attacks. More importantly, we develop a modeling and simulation integration platform that enables evaluation of resilience of CPS in the presence of cyber attacks based on attacker–defender games using simulations of sufficient fidelity.

The SecUre and REsilient Cyber–Physical Systems (SURE) platform incorporates 1) realistic models of cyber and physical components and their interactions; 2) cyber attack models that focus on the impact of attacks to CPS behavior and operation; and 3) operational scenarios that can be used for evaluation of cybersecurity risks. Further, it allows the evaluation of performance impact and assessment of resilient monitoring and control algorithms. The main innovation of our approach is that research processes and results are documented as executable software models, simulations, and generated data that support cybersecurity analysis and design in a quantifiable manner. A earlier version of the platform is demonstrated in [3]. The paper presents the platform using smart transportation systems as the CPS application domain. We evaluate the approach using three case studies: 1) vulnerability analysis of transportation networks to traffic signal tampering; 2) resilient sensor selection for forecasting traffic flow; and 3) decentralized resilient traffic signal control in the presence of denial-of-service (DoS) attacks. It should be noted that transportation systems are treated as any other network critical infrastructure, and hence, the proposed approach can be directly applied to other similar classes of CPS.

The SURE platform enables in-depth experimental evaluation of security and resilience that is necessary for developing the scientific foundations and technology. Theoretical analysis is accompanied by large amounts of experimental work and empirical observations use realistic CPS models and integrated simulations of tightly coupled cyber and physical components. Additionally, the platform allows the design and execution of controlled experiments of large-scale CPS by configuring the system and attack models. The main idea is to untangle poorly understood interactions and improve understanding by simulating real-world CPS. Simulation of such complex systems can lead to new knowledge by predicting how an assemblage of heterogeneous components will behave and discover what are the implications of the assumptions imposed on the system.

The rest of the paper is organized as follows. Section II provides a brief overview of secure and resilient CPS. Section III describes the goals and the system architecture of the SURE platform. Section IV presents the CPS modeling and simulation integration tools that are the main building blocks of the platform. Section V describes the main advances for adversarial modeling that is one of the most significant challenges for evaluation of CPS security and resilience. Section VI presents three case studies from transportation networks that illustrate the use of the platform. Finally, Section VII summarizes the main conclusions and discusses open research directions.

II. SECURE AND RESILIENT CPS

In recent years, a number of successful attacks against CPS targets, some of which have even caused severe physical damage, have demonstrated that security and resilience of CPS is a very critical problem. High-profile attacks have been reported in a broad range of CPS application domains. Stuxnet inflicted physical damage to an industrial infrastructure (uranium hexafluoride centrifuges) by attacking the supervisory control and data acquisition system (SCADA) [4]. The attack on Maroochy Water Services in Queensland, Australia disrupted pumping operations and suppressed alarms, resulting in the release of untreated sewage into local waterways [5]. Researchers have demonstrated the ability to compromise unmanned aerial vehicles (UAVs) [6]. Cyber attacks on modern automobiles that can lead to physical consequences, including disabling the brakes, killing the engine while the automobile is moving at high speed, permanently locking the doors, and manipulating the speed indicator have been reported [7], [8]. Attacks can target not only individual vehicles but transportation systems, for example, coordinated ramp metering attacks on freeway control systems are demonstrated in [9].

Because of its significance, security and resilience of CPS have attracted considerable attention in many application domains such as automobiles [10], medical devices [11], smart grids [12], industrial control systems [13], and transportation systems [14]. After performing vulnerability and risk analysis [15], typical security efforts aim at prevention of cyber attacks, intrusion detection, and resilient control design [16]. Because of the heterogeneity and complexity of these systems, methodologies that improve CPS security are very diverse with different objectives, specifications, and constraints resulting in a broad body of knowledge [17].

Prevention mechanisms aim to create technology to make CPS harder to attack successfully. Such efforts extend cybersecurity methods for information technology systems by taking into consideration salient features that include include long system lifecycles, restricted update capabilities, reliance on legacy systems and protocols, limited computational resources, and real-time constraints. The key challenge in such methods is the development of secure components, devices, individual systems, and networks that can be used in various CPS applications. Specific technologies include authentication [18], design of high-assurance software [19], formal methods for automated verification and testing [20], and architecture security [21].

Although preventive measures are necessary, it can be very expensive and even impossible to fully protect complex systems and a determined adversary can still launch successful attacks. Intrusion detection systems for CPS consider properties of the physical components to identify anomalous behavior that may be attributed to cyber attacks. Physical models are used for attack detection in [22]. Detection limitations for CPS and characterization of undetectable (or stealthy) attacks are presented in [23]. A game-theoretic approach for determine time-dependent thresholds to achieve an optimal tradeoff between detection delay and false positive rates is studied in [24]. A new metric to measure the impact of stealthy attacks and mitigation techniques by combining detection schemes are developed in [25].

Recently, there has been also great progress in resilient control design [26]–[29]. These methods are based on abstractions and models used for control design and complement methods that focus on software and network security. The secure system simplex architecture, for example, has been proposed to improve CPS security by detecting timing anomalies in the execution of control software caused by malware and switching to a trusted controller if malicious intrusions are detected [30]. Another class of methods of importance to many CPS focuses on secure state estimation in the presence of sensor attacks [31].

The value of developing testbeds for experimental evaluation of CPS security has been recognized and various testbeds have been proposed for security assessment of CPS application domains in critical infrastructure such as power systems and SCADA systems [32]–[35]. The PowerCyber CPS testbed, for example, is used to implement and evaluate cyber attacks on automatic generation control [36].

The SURE platform presented in this paper is developed for analysis of security and resilience of large-scale CPS such as transportation systems. Our main goal is to analyze the resilience of monitoring and control algorithms that operate in adversarial environments. The platform is built on top of a model-based modeling and simulation integration framework and provides the necessary tools and tool infrastructure to establish a coherent experimentation framework for evaluation of resilience. Performing controlled simulation experiments of large-scale CPS is valuable for developing realistic models, investigate poorly understood interactions, discover what are the implications of various assumptions, and predict how an heterogeneous assemblage of components will behave in the presence of cyber attacks.

III. GOALS AND SYSTEM ARCHITECTURE OVERVIEW

Our ultimate goal is to develop a systematic body of knowledge with both strong theoretical and empirical underpinnings to inform the engineering of secure and resilient CPS that can resist cyber attacks. The main idea is to perform CPS resilience studies based on attacker–defender games using simulations of sufficient fidelity. Consider, for example, the transportation network around the Vanderbilt University campus that is used throughout the paper to illustrate the approach. Such a transportation system consists of a large number of sensors and traffic lights (actuators) that communicate via networking technologies. This infrastructure provides valuable services such as traffic light control, congestion prediction and management that can be used for every day operation as well as planning of special events such as football games. The overall system behavior depends on physical components (vehicles), physical phenomena (traffic flow), cyber components (sensors, actuators, networks, control, monitoring algorithms), and humans who have diverse behaviors and reactions. Evaluating how resilient is the Vanderbilt campus transportation network in the presence of cyber attacks is a significant challenge that leads to multiple research questions. There is the need to 1) seek suitable metrics that can quantify resilience based on the services provided by the system; 2) capture diverse attacks such as traffic signal tampering, traffic flow sensor DoS, and integrity attacks as well as their effects in the system; 3) identify the most critical components (e.g., intersections) that need to be protected; 4) understand how traffic patterns affect the system behavior in the presence of attacks; and, of course, 5) design resilient monitoring and control algorithms that can provide acceptable levels of service even when the system is attacked.

Our objective is to provide a platform for experimentation and evaluation of resilience of CPS in the presence of cyber attacks. We formulate attacker–defender games, provide software tools that allow users to play the games, and perform extensive heterogeneous simulations for different player strategies to quantify resilience based on well-defined metrics. We express the utility of an attacker based on possible attacker's goals and we use the system and application requirements to quantify the utility of the defender. Based on these utility functions, we define metrics for resilience and prescribe ways to compute them using simulations. Finally, we design resilient monitoring and control algorithms and we evaluate them against various attack models.

In transportation networks, for example, the traffic signal schedule can be designed to minimize congestion. A typical goal of an attacker is to minimize the network's utility by maximizing congestion. An attacker may compromise the system by tampering with the schedule of the traffic signals in multiple intersections. Because of hardware fail-safes, the attacker will not be able to cause an accident, and further, to avoid detection, the attacker may select only valid schedules. In such a scenario, the SURE platform can be used to evaluate different attack and defense strategies by allowing users to configure attack and system models. In addition, we can compute the optimal strategies for the attacker–defender games and evaluate traffic congestion in the network assuming the worst case attack (given an attack model). Resilient algorithms for traffic light control that account for attacks can also be design and evaluated. Technical details of our approach and examples are presented in Section VI.

Possibly, the most significant challenge in analysis of CPS security and resilience is modeling cyber attacks. Real cyber attacks exploit software vulnerabilities to perform code injection or data tampering. Protecting the system by preventing such attacks may be very expensive or even impossible. In our approach, we assume attacks our successful and we focus on the impact of the compromised cyber components in the system. The attack models are inspired by reports of actual cyber attacks at vehicle detection systems embedded in roadways and traffic lights [37]. We develop abstracted models or various availability and integrity attacks and we include model attributes for representing constraints, attack start times and duration, and other features. Further, we develop a modeling language for composing complex and multistage attacks. The adversarial modeling language is presented in Section V.

Given the system and attack models, we automatically synthesize and perform integrated heterogeneous simulations that are used to translate abstract CPS security and resilience concepts into concrete observations. The SURE platform is based on a extensive software tool infrastructure for integrated CPS simulation and provides a suite of experimentation services for cloud deployment, collecting simulation results, and visualization capabilities. The software infrastructure for CPS modeling and simulation integration is presented in Section IV.

In summary, Fig. 1 provides an overview of the approach. The goal is to evaluate resilience of CPS in the presence of cyber attacks. The evaluation is based on attacker–defender games using simulations of sufficient fidelity. To rapidly synthesize complex heterogeneous simulations, we have developed a modeling integration framework and a tool infrastructure for attack and system modeling. The models are used by a model-based integration framework for heterogeneous and distributed simulations to support rapid design, synthesis, and evaluation of simulated experiments.

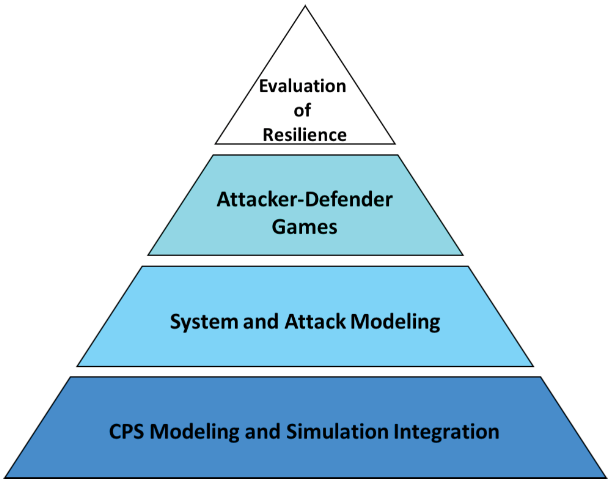

Evaluation
of
Resilience
Attacker-Defender
Games
System and Attack Modeling
CPS Modeling and Simulation Integration

Fig. 1. ***Goals and architecture overview.***

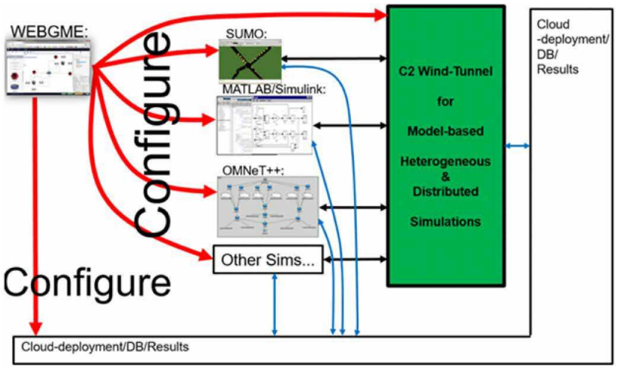

WEBGME:
SUMO:
MATLAB/Simulink:
OMNeT++:
Other Sims...
Configure
Configure
C2 Wind-Tunnel for Model-based Heterogeneous & Distributed Simulations
Cloud -deployment/ DB/ Results
Cloud-deployment/DB/Results

Fig. 2. ***SURE platform architecture.***

The primary focus of the SURE platform is to provide a flexible environment for evaluating the impact of various cyber attacks. The core challenge is to execute realistic and detailed simulation models while keeping the design and experimentation interface simple and focused on the security aspects. To achieve these goals, we use two core technology components: 1) the WebGME online modeling and collaboration tool [38]; and 2) the Command and Control Wind Tunnel (C2WT) simulation framework [39]. WebGME is a highly flexible online modeling environment built on state of the art web technologies, while C2WT can connect and synchronize established large-scale simulation tools using the high level architecture (HLA) interoperability standard [40]. For transportation systems, the simulation integration platform uses a modular approach to integrate the following simulation engines: OMNeT++ [41], SUMO [42], and Matlab/Simulink [43] as shown in Fig. 2. SURE provides the necessary domain-specific languages, models, model translation, and simulation driver tools to establish a coherent experimentation framework.

The main challenge is to provide an environment that provides mechanisms and tools for designing, deploying, and orchestrating experiments. We develop several mechanisms that hide the complexity of the tools and the integration environment, while presenting a highly user-friendly platform that allows focusing on CPS security and resilience without having to set up the underlying experimentation and computation infrastructure. The main interface to the platform is provided by the WebGME modeling environment. In the case of transportation networks, WebGME provides tools and methods to model traffic maps, traffic demand models, sensors, traffic lights, the communication network topology as well as an attack modeling language. As shown in Fig. 2, these models are used to configure not only the integrated simulation in C2WT, but also the experimentation infrastructure, which currently is supported for the OpenStack cloud and the Amazon AWS cloud. Additionally, a set of simulation drivers are configured to send experiment setup files and data to C2WT, monitor the experiments, gather experimental results, and visualize the data.

An important feature of the SURE platform is that it allows model-based design of controlled experiments for evaluation of resilience in CPS. In the transportation domain, system models (e.g., location and type of sensors, traffic light control algorithms, network topology, and protocols) and attack models (e.g., integrity and DoS attacks) can be easily configured. In addition, the models can be used to realize different traffic conditions that can greatly impact the results (e.g., the effect of the same attack can be different during morning rush hour and afternoon rush hour).

IV. CPS MODELING AND SIMULATION INTEGRATION

A. Domain Modeling for Rapid Experimentation

The underlying idea with WebGME and its predecessor, the generic modeling environment (GME) [44] is to facilitate the design of domain specific graphical modeling languages (DSMLs) for a wide spectrum of engineering and science domains. A DSML abstracts the commonalities in the domain while the models capture the information specific to the given system being modeled. The designer of the language has a high degree of freedom to decide which are the important first-class concepts of the domain and how to capture such elements visually. Once the language is defined, concrete models can be built to analyze the system, provide input to simulators, generate test cases, and create documentation. This DSML-based technique provides guaranteed consistency because every tool utilizes the same set of shared models and visual syntax. Tool suites built around WebGME and GME had been applied successfully in multiple domains [45]–[51].

WebGME provides a unique approach for capturing the concepts and rules of the domain. The visual language is described and embedded with the model—this partition is called the metamodel. Due to this tight integration and unified representation of the language and its instances, the environment allows to make changes to both during the evolution of the model. In the SURE platform, we heavily depend on this capability to continuously extend and prune the DSML while integrating new CPS application domains and simulation tools, and developing new security scenarios. Using this incremental approach, we can keep the modeling language and the user interface concise. This is especially important in the development and refinement of the adversarial modeling language which is the most challenging part of the metamodel for evaluating security and resilience. Beyond capturing and enforcing the rules of the metamodel, WebGME uses two first-class modeling concepts to build models with increasing levels of abstraction. Hierarchical decomposition is the most widely used technique to handle complexity. Copying, moving, or deleting a model will copy, move, or delete its constituent parts. In the SURE platform, this feature allows us to create refined models of physical or cyber entities but handle these as single components at higher levels in the hierarchy. Prototypical inheritance is a unique feature that enables the modeler reuse and refine models. Just as there is a single composition tree, there is a single inheritance tree with a single object at its root. Rules specified by the metamodel, as well as actual model parts, propagate down this tree. Deleting a model will delete all of its descendants in the inheritance hierarchy too. The SURE platform builds on this capability for creating libraries of reusable components for the CPS domains and attack models.

The current metamodel of the SURE platform is partitioned to three distinct areas: 1) a high-level model of the cyber infrastructure using common network abstractions such as routers, network links, and end points; 2) a rich toolset for developing attack models (details are presented in Section V); and 3) a set of elements for integration with a concrete CPS domain. Due to the relative separation of these concepts, we can add new physical domains with minimal changes to the cyber and security aspects. Such changes include capturing the essential physical domain concepts (e.g., sensors, actuators) and their linkage to the elements of the cyber infrastructure. The separation allows the development of relatively simple simulation drivers which translate the model elements to input specifications for the application-specific simulation engine(s).

Within a specific CPS domain (e.g., transportation networks), the platform supports multiple security scenarios. A scenario consists of a concrete model in the physical domain, predefined elements in the cyber domain, and most importantly, a specific challenge problem to be investigated. The challenge problem focuses on a well-defined application (e.g., traffic forecasting or traffic signal control) and defines the goal and metrics for the evaluation. Each scenario requires the development of analysis tool plugins for processing the results of the simulation.

B. Collaboration and Gamification

The SURE platform uses a web browser-based user interface and supports online collaboration where changes are immediately broadcast to all users. This is similar to how live collaboration tools (e.g., GoogleDocs) work, except here the shared artifacts have a much richer data model which makes consistency management more challenging. The platform persists each security scenario model on a centralized server, thus these are accessible from anywhere at any time using a web browser. Also, the models can be opened and edited by multiple users at the same time. Concurrent editing conflicts are detected, retried, or rejected with immediate visual feedback. The web-based interface also provides easy access to the simulation and analysis tools for all connected users.

The multiuser access is especially important for developing scenarios based on attacker–defender games. In these models, each client is assigned to a group of defenders (blue team) or attackers (red team). The scenario defines the rules of engagement, that is the set of cyber elements which can be attacked, the tools for defending against attacks, and the goals of the attacker and defender teams. The modeling environment allows the blue and red teams to work on the scenario model concurrently or sequentially. The integrated scenario and simulation drivers allow to evaluate the performance of both teams repeatedly. The simulation results are provided to the users through the same web-based interface (see Section VI for case studies).

C. Traceability and Reproducibility

One of the most critical and often overlooked problems of experimentation-based evaluation platforms is the precise and accurate preservation of both the inputs and the results. Typically, such bookkeeping is the sole responsibility of the experimenter and requires diligent archiving of each step. Such approaches breakdown fast in complex systems and significantly burden the experimentation process. These problems are even more critical in a collaborative environment with multiple users and shared responsibilities. Reproducibility, traceability, sharing, and trust in experiment-based evaluation require support mechanisms.

The SURE platform includes a strict but highly transparent versioning mechanism for all artifacts of the scenario model and the results of simulation runs. Further, its cloud-based architecture enables to automatically capture provenance information. Users can access the tools using their browsers, thus almost all artifacts related to the experiments can be controlled using the web-based infrastructure. The internal storage model is heavily influenced by the Git distributed version control system [52]. Every change to the model is recorded with an automatically commit object and is protected by a cryptographic hash. The commit objects record the modifications and the previous commit(s), to which these changes has been made. Thus, every project can be traced back to its origin. The model also allows separate development branches to be created and merged back. Specific stages in the tree of the history can be tagged and returned.

Note that some artifacts, most notably the outputs of the simulation runs, cannot be conveniently stored as WebGME models. For versioning and tracking such external resources, WebGME includes the Asset Manager service. This storage service provides a convenient way to store and retrieve unstructured files in the cloud similarly to Amazon S3 [53] or OpenStack Swift. The storage engine keeps all previous versions of the files and WebGME maintains external references as built-in first-class concepts to the external artifacts. Since the references are part of the core WebGME data model, the version dependencies between the external artifacts and the model are captured as well. These references are important elements of the provenance information.

D. Transportation Domain

The current version of the SURE platform is used to evaluate resilience of CPS transportation systems. The simulation capabilities in this domain are provided by Simulation of Urban MObility (SUMO) [42], a microscopic, intermodal and multimodal, space-continuous and time-discrete traffic flow simulation platform. SUMO uses arbitrary street networks down to individual lanes and detailed intersection models. It simulates the traffic flow at the vehicle level using a realistic behavior model of each driver. Complex traffic rules and signaled intersections are also supported. The two most important simulation inputs are the street network and a parameterization of the traffic demand which can be defined using departure locations and times with intended destinations or complete route specification.

SUMO allows arbitrary maps to be created and used. Fig. 3 shows the transportation network around the Vanderbilt University campus. SUMO provides auxiliary tools for importing regions from OpenStreetMap [54] into its native XML-based representation. We also added a custom visualization tool to WebGME, which can render these SUMO maps on the model canvas. Note that the detailed street network is not part of the versioned model but and external artifact. Therefore, it is important to create unambiguous links to the entities of this map from first-class model elements. The custom visualizer tool is responsible for this data association problem. When DSML entities (e.g., routers and controllers) are created or moved on the map, the visualizer automatically updates linking attributes based on the location and proximity to rendered SUMO map elements (e.g., lane or intersection).

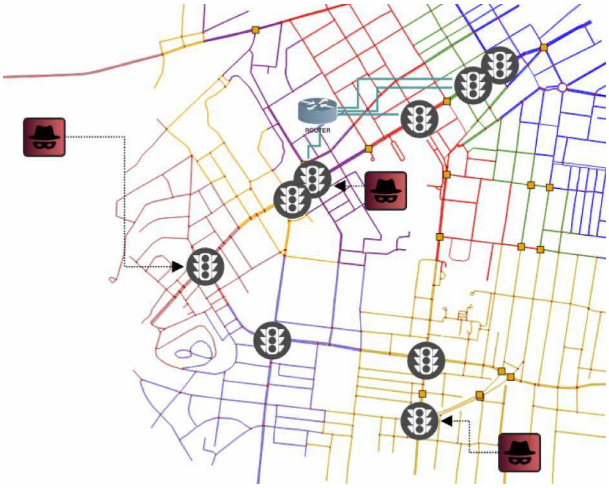


Fig. 3. ***Resilient hierarchical control scenario using the Vanderbilt campus map.***

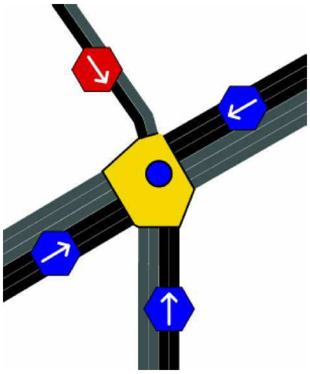


Fig. 4. ***Hierarchical refinement: Intersection-level model.***

In simple operational scenarios, model elements are created and linked only at the top level on the global map overlay. However, more detailed models require hierarchical decomposition. In transportation systems, such deeper hierarchies are created, for example, at road intersections. Fig. 4 shows the model of one intersection. Note that some of the visual elements (e.g., lanes and the shape of the intersection) are still rendered virtually using the external SUMO map resources.

A typical exploration in transportation systems is to evaluate the same attacker–defender games with different traffic patterns. We add a simple sublanguage for capturing stochastic traffic demand parameters. Each traffic pattern is represented by a state with a predefined duration and the probabilities of car departures and destinations. Different probabilities can be assigned based on the departure (edge or inside) and destination locations. The model can be used to simulate morning and afternoon peak commuter traffic, busy midday hours, and quiet weekends or nights. Entire programs defining traffic demand patterns can be stored in the library as part of the project and applied to a scenario model effortlessly.

The platform uses the following DSML concepts for designing system and attack models: 1) intersections as container models to start new model hierarchies; 2) sensors which correspond to induction-loops or multientry/multiexit sensor devices for measuring line crossings or occupancy; 3) traffic lights which are entities within an intersection and capture traffic lights and their schedules; 4) local controllers that receive inputs from the intersection-level sensors and control the timing of the traffic light schedule; 5) regional controllers which are high-level controllers responsible for optimizing the global performance of the transportation network; and 6) traffic demand and demand state for modeling traffic patterns as described above.

In addition to the transportation-specific model elements, the scenario models contain routers, network links, and base stations. These elements are used to describe the communication network and are possible targets of cyber attacks (in addition to the transportation-specific elements).

E. Simulation Drivers

The reconfigurable web-based modeling interface captures and stores the system model and the design decisions made by the red or blue teams. The power of the SURE platform comes from its integration with existing large-scale simulation engines in the backend infrastructure. The platform implements a multistage translation process to leverage such simulators (e.g., OMNeT++, SUMO, and Matlab/Simulink) without building fragile point solutions. The architecture is shown in Fig. 5.

The detailed timing and data type-level integration challenges are handled by the C2WT platform using the HLA standard. C2WT and the integrated simulation engines are deployed on dedicated cloud server instance(s) and form a self-contained independent service. The platform allows complete simulation runs to be executed without any dependencies on the WebGME front-end. Extending the SURE platform for experimentation in new CPS application domains requires integration of new domain-specific simulation engines in the C2WT architecture.

Executing C2WT-based simulations from the modeling environment requires a few model transformation steps. First, the entire graphical scenario model is traversed and its content is exported to a JSON file [55]. This step is independent of the CPS domain and the actual scenario. The next stage is responsible to filter, check, and transform the generic data based on the goals of the selected scenario. This layer is also responsible for transforming low-level simulation results to scenario-specific metrics at the end of the simulation. The same or similar scenario can potentially be used in different CPS application domains. Finally, the simulation drivers prepare the input files for the C2WT-based experiment. This transformation is highly specific to the simulators used in the particular CPS domain. The drivers are also responsible for the completion of the experiment and the collection of all relevant outputs of the simulators. This multistage process allows us to reuse many existing components and adapt to new CPS domains to the SURE platform without significant changes to the current architecture.

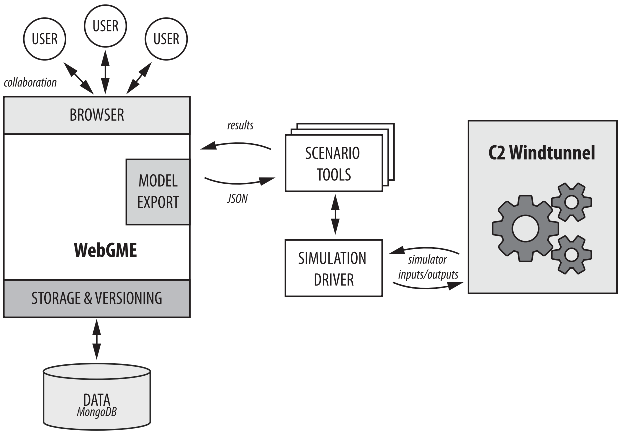


Fig. 5. ***Integrating the web-based testbed interface with simulation backend using scenario analysis tools and simulation drivers.***

F. Model-Based Simulation Integration

The C2WT is a model-based integration framework for heterogeneous and distributed simulations which supports rapid design, synthesis, and evaluation of distributed simulations [39], [56]. The framework provides an intuitive and extensible platform for rapidly composing integrated simulations using a variety of special purpose tools that span many CPS domains. C2WT provides a modeling language and tools to build an integration model of a system of systems. In the SURE platform, these involve creating an integration model to compose simulations of the transportation network, the communication network, various controllers, and several other experiment-specific modules. The platform supports automatic synthesis of runtime adapters, artifacts, and execution scripts that are needed to compose and execute integrated simulations. In addition, it supports modeling the deployment of simulations on a cluster so that individual simulations can be deployed automatically at runtime and monitored through its simulation management component. Fig. 6 shows the architecture diagram of the C2WT.

The C2WT framework provides a HLA-based simulation integration platform [40]. The HLA implementation is called runtime infrastructure (RTI) and C2WT relies on an open-source implementation called Portico [57]. The RTI enables integrated and synchronized simulation of a number of federate types (individual simulation components). Over the last decade, the C2WT has developed into a highly mature framework with support for a variety of simulation tools that can be easily integrated such as Matlab/Simulink [43], OMNeT++ [41], CPNTools [58], SUMO [59], TrainDirector [60], and Gridlab-D [61] among others. Apart from these special-purpose simulators, C2WT supports integration of generic simulations written in C++ and Java programming languages. In SURE, we leverage the integration support of SUMO for traffic simulation, OMNeT++ for

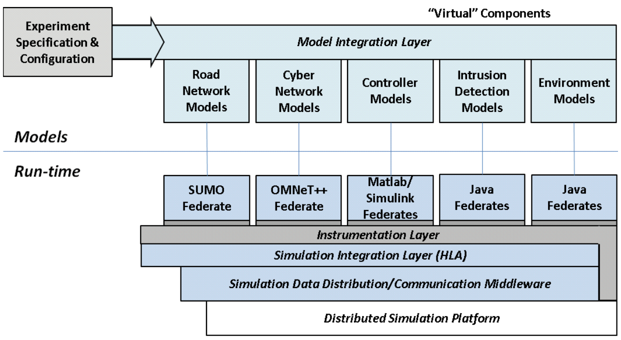

Experiment Specification & Configuration
"Virtual" Components
Model Integration Layer
Road Network Models
Cyber Network Models
Controller Models
Intrusion Detection Models
Environment Models
Models
Run-time
SUMO Federate
OMNeT++ Federate
Matlab/ Simulink Federates
Java Federates
Java Federates
Instrumentation Layer
Simulation Integration Layer (HLA)
Simulation Data Distribution/Communication Middleware
Distributed Simulation Platform

Fig. 6. ***C2WT simulation integration framework.***

communication network simulation, and Matlab/Simulink, C++, and Java for various types of controllers and other scenario-specific modules.

A core module of C2WT is the model integration layer (MIL). The MIL comprises a graphical modeling language that allows creation of domain-specific integration models. The language allows defining data models specific to scenarios, creating abstract models for the core simulation components, and defining the data flows relating the core component models with the data models. In addition, it supports modeling of the experiments, the computation hardware infrastructure, the configuration of experiments, and the deployment of simulations. The MIL also provides a number of domain-specific synthesis tools that allow automatic synthesis of customized wrappers that facilitate HLA-based integration of various simulation models. Also, several synthesis tools enable the automatic experiment configuration and deployment.

C2WT employs several models for a particular scenario. These models specify the simulation components, the external inputs given to the simulation components at runtime, and the runtime, logging, and control configurations. Also, C2WT allows the modeler to define the set of computers where the experiment can be executed along with login credentials. The user can specify the mapping of simulation tools (federates) to the computer where they need to be executed. C2WT automatically generates appropriate shell scripts and configuration to deploy the simulation to configured computers to be able to execute the entire integrated simulation.

V. MODELING CPS IN ADVERSARIAL ENVIRONMENTS

The primary goal of adversarial modeling is to be able to describe sophisticated executable attack strategies. The purpose of the simulation-based experiments is to analyze the effects of cyber attacks on the physical domain. The SURE platform provides a concise set of generic well-understood attack mechanisms. However, the actual execution of each attack requires intricate integration in the CPS model. We develop an extensive cyber attack library and implement the mechanisms for integrating the attacks with the system model. Complex, multistage attacks are implemented using courses of actions (COAs). We incorporate the COAs in the SURE metamodel using a well-defined sublanguage for capturing adversarial attacks. Based on this visual language and using prototypical inheritance in WebGME, users can develop sophisticated and coordinated attacks. Currently, the targets of attacks are restricted to the cyber domain.

A. C2WT Cyber Attack Library

We develop a reusable and modular cyber attack library as part of the C2WT. Fig. 7 shows the main cyber attacks available in this library that can be used to configure attacks like distributed DoS, network delays, data corruption, network manipulation, and others. Configuration parameters for the attacks are also shown in the figure. For example, consider an integrity attack deployed on a network node in the communication network model. The attack manipulates the network packets flowing through the network node at the message level. The parameters are used to change different fields of the message while being consistent to their data types.

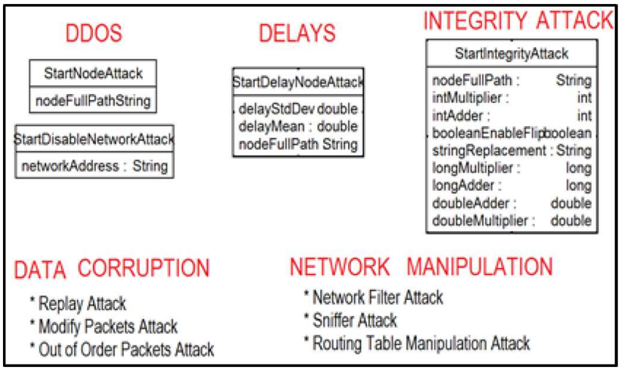


Fig. 7. ***C2WT cyber attack library.***

Fig. 8 shows part of the integration model of the communication network simulation component in C2WT represented by the (green) federate box labeled omnet. This component (referred to as omnet federate) is developed using the simulation tool OMNeT++ and the model suite INET which provides support for multiple Internet protocols [62]. This generic federate component understands messages of type NetworkPacket which contain the routing information in the network topology, metadata of the message, and the payload itself. The figure also shows models for DoS and integrity attacks. Multiple cyber attacks are designed, implemented, and supported by the library. These

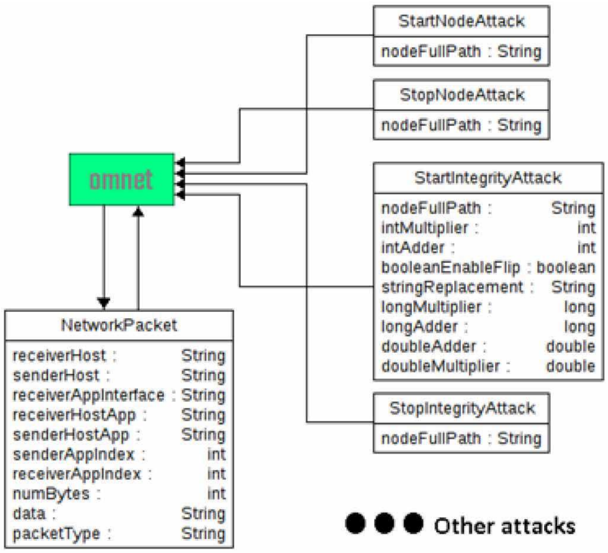
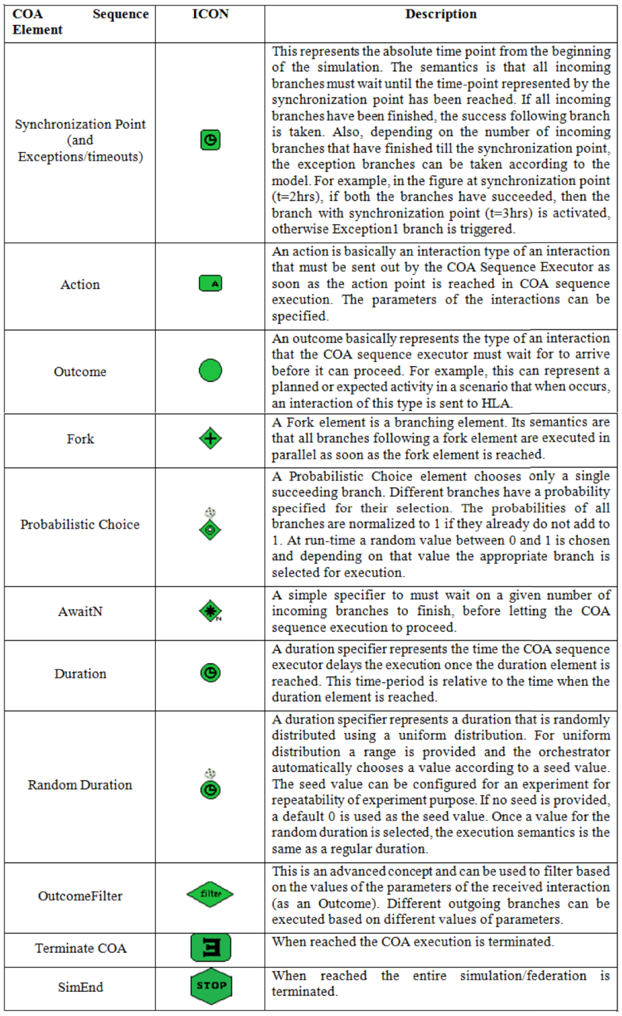


Fig. 8. ***C2WT communication network component and cyber attack library integration.***

are atomic attacks that can be deployed on any node in the communication network and they are modular and configurable for extending and parameterizing them to support various security scenarios.

In Fig. 8, the omnet federate both publishes and subscribes to NetworkPacket interaction type. Any communication path between two federates can be relayed via a simulated communication network with the appropriate network topology, routing, and protocols. In order to support this interaction type in the network simulation component, a separate module is needed to translate regular interactions into NetworkPacket and back. When a federate sends a message that is to be transmitted over a simulation network, first the message is translated into a NetworkPacket and then sent to the omnet federate. The omnet federate routes the data in the communication network and when it is received at the network node corresponding to the recipient federate, it is translated back to the original interaction type. C2WT supports many different types of mapping specification between interactions that allow automatic message translation and even data type conversion. Further, the mappings support one-to-one, one-to-many, many-to-one, and many-to-many specifications.



COA Sequence Element	ICON	Description
Synchronization Point (and Exceptions/timeouts)	◷	This represents the absolute time point from the beginning of the simulation. The semantics is that all incoming branches must wait until the time-point represented by the synchronization point has been reached. If all incoming branches have been finished, the success following branch is taken. Also, depending on the number of incoming branches that have finished till the synchronization point, the exception branches can be taken according to the model. For example, in the figure at synchronization point (t=2hrs), if both the branches have succeeded, then the branch with synchronization point (t=3hrs) is activated, otherwise Exception1 branch is triggered.
Action	A	An action is basically an interaction type of an interaction that must be sent out by the COA Sequence Executor as soon as the action point is reached in COA sequence execution. The parameters of the interactions can be specified.
Outcome	●	An outcome basically represents the type of an interaction that the COA sequence executor must wait for to arrive before it can proceed. For example, this can represent a planned or expected activity in a scenario that when occurs, an interaction of this type is sent to HLA.
Fork	✚	A Fork element is a branching element. Its semantics are that all branches following a fork element are executed in parallel as soon as the fork element is reached.
Probabilistic Choice	◈	A Probabilistic Choice element chooses only a single succeeding branch. Different branches have a probability specified for their selection. The probabilities of all branches are normalized to 1 if they already do not add to 1. At run-time a random value between 0 and 1 is chosen and depending on that value the appropriate branch is selected for execution.
AwaitN	✱ N	A simple specifier to must wait on a given number of incoming branches to finish, before letting the COA sequence execution to proceed.
Duration	◷	A duration specifier represents the time the COA sequence executor delays the execution once the duration element is reached. This time-period is relative to the time when the duration element is reached.
Random Duration	◷	A duration specifier represents a duration that is randomly distributed using a uniform distribution. For uniform distribution a range is provided and the orchestrator automatically chooses a value according to a seed value. The seed value can be configured for an experiment for repeatability of experiment purpose. If no seed is provided, a default 0 is used as the seed value. Once a value for the random duration is selected, the execution semantics is the same as a regular duration.
OutcomeFilter	filter	This is an advanced concept and can be used to filter based on the values of the parameters of the received interaction (as an Outcome). Different outgoing branches can be executed based on different values of parameters.
Terminate COA	Ǝ	When reached the COA execution is terminated.
SimEnd	STOP	When reached the entire simulation/federation is terminated.

Fig. 9. ***COA sequence elements in C2WT.***

B. Courses of Actions for Multistage Attacks

In order to rapidly evaluate complex and multistage attacks, we develop in C2WT a capability to model, configure, and evaluate COAs. COAs represent sequence of triggers and actions that can be combined using workflows. COAs utilize atomic actions based on triggers such as time, system events, or realized outputs generated at runtime during the simulation. Fig. 9 lists the elements that can be used in construction of COAs and a description of their behavior. Each of the COA elements has a set of parameters that can be used to configure its intended behavior.

An important capability of the language is that it allows grouping a set of related COAs into COA groups. C2WT provides mechanisms to specify how a particular COA from a COA group should be chosen in an experiment run. C2WT provides tools, GUIs, database, and file logging support for executing these combinations of experiments in a batch mode so that all the simulation results are collected and organized for analysis. Using COA groups, it is possible to evaluate system resilience under a large number of alternatives in an efficient manner.

C. Adversarial Modeling Language

The adversarial modeling language available to the user using the web-based SURE interface is based on the C2WT attack library and the services for executing COAs. Elements of the COA model are distilled into a visual language shown in Fig. 10. These elements are grouped in two categories: attack actions are executed against the cyber components of the model, while triggers are used to coordinate the execution of actions. Fig. 11 shows a typical attack model with the two types of blocks interleaved to form an alternating pattern.

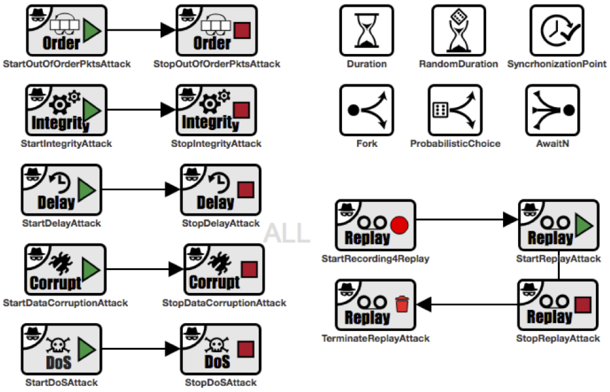


Fig. 10. ***Elements of the adversarial modeling language.***

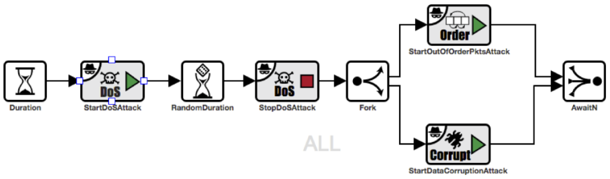


Fig. 11. ***Example adversarial model: executing multiple attacks in sequence and in parallel.***

Attack types supported by the adversarial modeling language include the following.

1) *Out of order:* This attack targets a node or a router and results in rearranging the network packets passing through.
2) *Integrity:* This attack modifies the content of the packets.
3) *Delay:* This attack introduces probabilistic network delays using configurable parameters (mean and variance).
4) *Data corruption:* It is a random corruption of data packets.
5) *DoS:* This attack prevents packets to be received or transmitted.
6) *Record and playback:* This attack records and replays a set of data.

Each attack type has separate attributes to initiate and to stop the execution of the attack.

The following triggers are used to implement the control flow. 1) Duration: A time-based delay for executing the next block—both deterministic and probabilistic intervals are supported. 2) Forking and branching: The execution follows in multiple threads in parallel or in one of the multiple branches. 3) Synchronization: Waiting for multiple threads of execution (created by the Fork trigger) or waiting for a global time point.

The example model in Fig. 11 deploys multiple attacks in sequence against its target node. After a deterministic delay, it executes a DoS attack for a random time duration. After this step, it triggers a data corruption corruption attack in the node and a packet reordering attack across the stream of packets passing through.

Each attack model is transformed to the native XML representation of COA scripts and packaged with other input artifacts before executing the C2WT simulation. The model transformation tool and the COA engine is generic and can be reused across various scenarios and physical domains.

VI. CASE STUDIES

Transportation systems are vulnerable to cyber attacks that include communication network attacks such as DoS and integrity attacks such as data corruption. Regardless of the specific nature of the attack, the objective of the adversary is to degrade the quality of human decision making by limiting access to information, by decreasing trust in information resources, by modifying important information, or by tampering with applications and application results. We present three case studies to illustrate the capabilities of the SURE platform. The first case study presents an approach for vulnerability analysis of transportation networks to traffic signal tampering attacks. In this case, we focus only on the attacker, we provide an algorithm for computing the worst case attack, and we analyze its impact to the traffic flow using total travel time as metric. The second case study presents an approach for resilient sensor selection for traffic forecasting. We present an attacker–defended game for selecting the locations of sensor devices so that placement is resilient to DoS attacks that aim to degrade prediction accuracy. The simulations using the SURE platform are valuable to evaluate how resilient is the sensor selection approach under different traffic conditions. The third case study presents an approach for resilient traffic signal control where we demonstrate how the SURE platform can be used successfully to systematically and efficiently design and analyze the resilience of multiple-intersection closed-loop traffic light control. It should be noted that these case studies represent experiment instances but the SURE platform can be configured to evaluate resilience under different attacks, traffic patterns, and monitoring and control algorithms.

Fig. 12 shows the main steps for designing and executing experiments. The first step is to select the transportation network of a particular geographical region, and download the map (e.g., from openstreetmap.org). The map is imported into SUMO using the tool NETCONVERT. Once the road map is created, various traffic demand patterns are developed to support the designed experiments. We have created a graphical modeling tool for defining different (stochastic) traffic patterns as described in Section IV. In addition to the transportation network, the communication network needs to be modeled including sensors, traffic lights, routers, and base stations. Fig. 13 shows part of a model. During the experiment setup, we can create multiple copies of the model that correspond to different configurations that

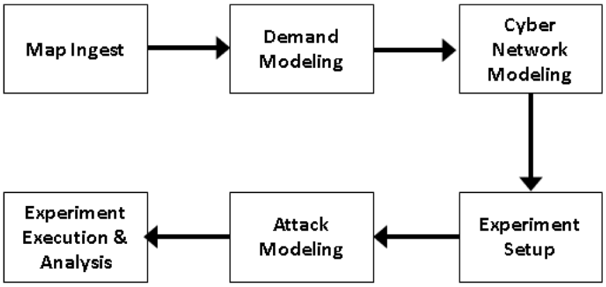


Fig. 12. ***Experimental workflow for creating new scenarios in SURE.***

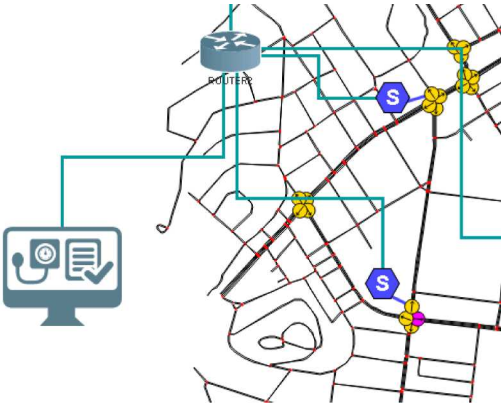


Fig. 13. *Cyber communication network modeling in SURE.*

include only the elements of interest. The attacks are modeled using the adversarial modeling language presented in Section V.

Finally, we execute the experiments using the available execution plugins. The plugins gather the appropriate modeling and configuration details and package them in a format suitable for integrated cosimulation in the C2WT. Execution progress is shown through status updates via notifications. When the simulation is completed, the results are processed and shown to the user.

A. Vulnerability Analysis to Traffic Signal Tampering

The evolution of traffic signals from standalone hardware devices to complex networked systems has exposed them to cyber attacks. While traditional hardware systems are susceptible only to attacks based on direct physical access, modern systems are vulnerable to attacks through wireless interfaces or even to remote attacks through the Internet. A recent case study analyzed the security of traffic infrastructure in cooperation with a road agency located in Michigan [37]. The agency operates around a hundred traffic signals, which are all part of the same wireless network, but the signals at every intersection operate independently of the other intersections. The study found three major weaknesses in the traffic infrastructure: 1) lack of encryption in the network; 2) lack of secure authentication due to the use of default usernames and passwords on the devices; and 3) vulnerabilities to known exploits. Even if every weakness discovered by such an investigation were corrected, it is extremely difficult to prevent all future software vulnerabilities. In addition to the general difficulty of this task, traffic signals pose further challenges such as long system lifetime and complicated software upgrade procedures. Consequently, ensuring that there will not be any opportunities for attack during the lifetime of a system is practically impossible, and we must consider the impact of successful attacks.

Due to hardware based fail-safes, compromising a traffic signal does not typically enable an attacker to switch the signal into an unsafe configuration, which could lead to traffic accidents. However, compromising a signal does enable tampering with its schedule, which allows an attacker to cause disastrous traffic congestions. In order to increase the resilience of transportation networks to tampering attacks, we must first be able to assess how vulnerable a given network is, that is, we must be able to estimate the potential impact of tampering attacks. Since this impact depends on the transportation network, the configuration of the uncompromised signals, as well as the traffic in a nontrivial way, vulnerability assessment is a challenging problem.

We propose an approach for evaluating the vulnerability of a transportation network to traffic signal tampering attacks. We develop a system and an attack model and we formulate the problem of finding worst-case tampering attacks. The problem of finding the worst case attack is computationally challenging, and we introduce an efficient heuristic algorithm for practical application. The theoretical approach is based on Daganzo's cell transmission model [63], a widely used macroscopic traffic model. Details of our approach can be found in [64]. Here, in contrast, we use the microscopic traffic model provided by SUMO and we utilize the SURE platform to evaluate the approach by computing the effect of the attack in the total travel time under various traffic conditions. Note that although we present results only for the worst case attack given the attack model, the SURE platform allows similar analysis to be performed for other attacks specified by the user.

1) System Model: We first introduce the traffic and attacker models used by our approach and then formalize how to quantify the vulnerability of a transportation network.

a) Traffic model: We let the set of intersections in the transportation network denoted by $\mathcal{S}$. For each intersection $s \in \mathcal{S}$, we let $\Gamma^{-1}(s)$ denote the set of incoming roads. For a given incoming road $k \in \Gamma^{-1}(s)$, we let p_{ks} denote the fraction of time that vehicles arriving from this road are allowed to pass through the intersection, which is determined by the schedule of the traffic signal. The resulting traffic flows can be determined by a microscopic (e.g., SUMO) or macroscopic traffic model (e.g., Daganzo's cell transmission model [63]).

b) Attacker model: Next, we introduce the attacker model which defines the attacker's action space and goal. We model attackers who can compromise some of the traffic signals and tamper with their configuration (i.e., schedule), thereby dramatically increasing the total travel time.

Action Space: The attacker's strategic choice is to select: 1) a set of traffic signals to compromise; and 2) a new schedule for each one of the compromised signals. We assume that the attacker is resource bounded, which means that it can compromise at most B intersections at the same time. Further, we assume that hardware-based fail-safe prevents the attacker from selecting an invalid schedule, such as setting both lights to green for two intersecting directions.

Goal: We assume a worst-case attacker whose goal is to minimize the network's utility, that is, to maximize the total travel time. For a given attack $\mathcal{A}$ (i.e., selection and reconfiguration of signals at most B intersections), let us denote by $T(\mathcal{A})$ the total travel time computed from the traffic model for the attacked network. Then, we can express the problem of finding a worst case attack as

$$\max_{\mathcal{A}} T(\mathcal{A}). \tag{1}$$

c) Vulnerability metric: We can define the vulnerability of a transportation network to traffic signal tampering attacks in an intuitive way as

$$\frac{T(\mathcal{A}) - T}{T} \tag{2}$$

where $\mathcal{A}$ is the worst case attack given by our attacker model and T is the total travel time of the network with the default configurations of the traffic signals.

2) Heuristic Algorithm: To quantify the vulnerability of a transportation network, we have to find a worst case tampering attack. This problem is computationally challenging since the number of different attacks to choose from grows exponentially with the size of the problem instance. Indeed, we have shown that finding a worst case attack is an NP-hard problem if we use Daganzo's cell transmission model as the traffic model [64]. Consequently, we introduce an efficient heuristic algorithm (see Algorithm 1) for finding near worst case attacks in practice.

Algorithm 1 Polynomial-Time Heuristic Algorithm for Finding an Attack

```
$\mathcal{A} \leftarrow \varnothing$
for $b = 1, \dots, B$ do
    for $s \in \mathcal{S}$ do
        for $k \in \Gamma^{-1}(s)$ do
            $\mathcal{A}' \leftarrow \mathcal{A} + (s, \{\hat{p}_{ks} = 1, \forall j \neq k : \hat{p}_{js} = 0\})$
            if $T(\mathcal{A}') \geq T(\mathcal{A}^*)$ then
                $\mathcal{A}^* \leftarrow \mathcal{A}'$
            end if
        end for
    end for
    $\mathcal{A} \leftarrow \mathcal{A}^*$
end for
Output $\mathcal{A}$
```

3) Simulation Results: We evaluate our approach using the SURE platform. We select five major intersections around the Vanderbilt campus as possible targets $\mathcal{S}$ for an attack (marked by red disks in Fig. 14). We consider four traffic demand patterns: morning commute, midday traffic, afternoon commute, and nighttime traffic. The schedules of the traffic signals are selected to minimize travel time without an attack.

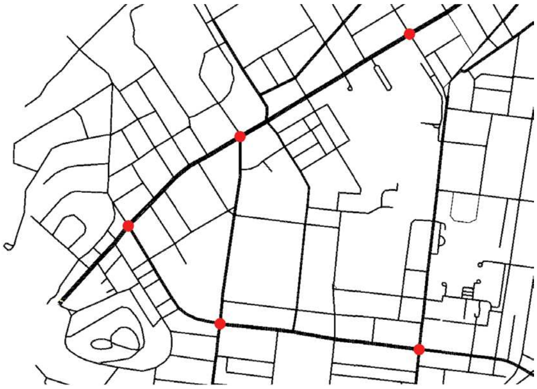


Fig. 14. ***Vulnerability analysis scenario; possible targets for an attack are marked by red disks.***

a) Varying traffic conditions: Fig. 15 shows the travel times with heuristic attack and without attack for various traffic scenarios. In this experiment, we fix the attacker's budget to $B = 3$. The figure shows that the vulnerability of the transportation network varies between 51% (midday scenario) and 92% (morning scenario).

b) Varying attacker budget: Fig. 16 shows the travel times resulting from attacks found by the heuristic algorithm and by exhaustive search in the afternoon traffic scenario. The figure shows that the heuristic algorithm performs exceptionally well, the difference being less than 0.8% to the exhaustive search in terms of the resulting travel time.

B. Resilient Sensor Selection for Traffic Forecasting

The ability to control any system hinges on having accurate information about its evolving state, obtained through persistent system monitoring. In many applications, such as transportation networks, the system to be monitored can extend over a large area, with many possible points of

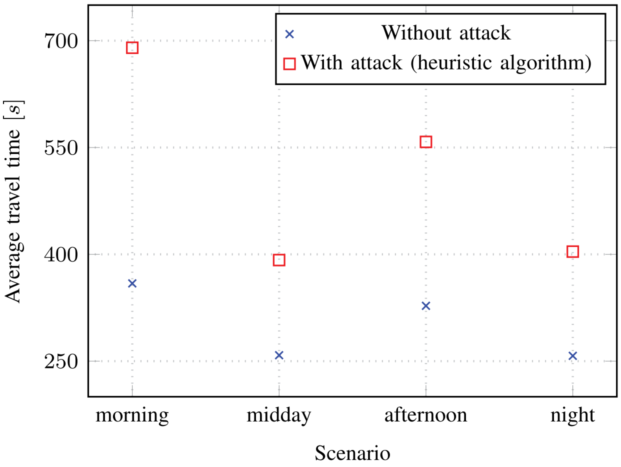


Fig. 15. ***Travel times with heuristic attack and without attack for various traffic scenarios in the vulnerability analysis case study.***

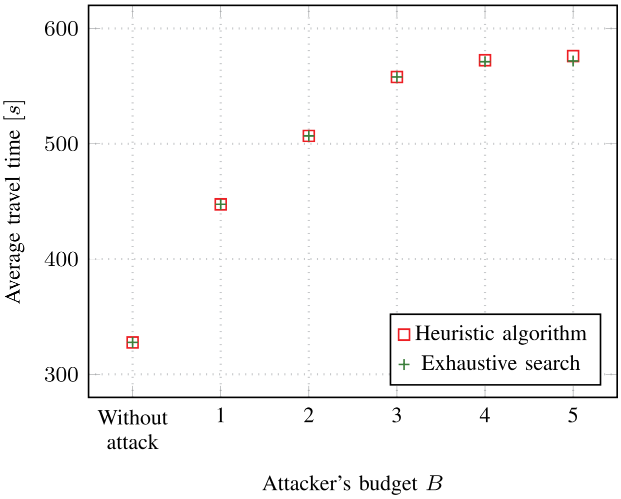


Fig. 16. ***Travel times resulting from attacks found by the heuristic algorithm and by exhaustive search in the afternoon scenario of the vulnerability analysis case study.***

observation. Although these areas can be very large, the number of sensors that can be deployed is limited by financial and/or technological constraints. Consequently, we are faced with a problem of finding locations for placing a limited number of sensors so as to minimize our posterior uncertainty about the quantities being monitored. Due to its importance, this problem of sensor placement (or, more generally, observation/feature selection) and associated predictions about unobserved state variables has received considerable attention particularly when variables of interest are modeled using a Gaussian process regression [65]. For example, Gaussian process regression models have been successfully applied to a wide range of problems, such as traffic volume forecasting [66], [67], wind power forecasting [68], estimation of water chlorophyll concentration [69], and spectrum sensing [70].

We introduce an approach for selecting the locations of sensor devices so that placement is resilient to DoS attacks that aim to degrade prediction accuracy. The approach is presented in detail in [71] where it is evaluated using a single data set of real-world traffic measurements. In contrast, in this paper, we performed multiple controlled simulation experiments and analyze the approach using various traffic conditions in the Vanderbilt campus transportation network. First, we use the SURE platform to estimate the parameters of a Gaussian process based regression model. Since finding an optimal selection for the sensors is an NP-hard problem, we propose a heuristic algorithm, which we show to be efficient using numerical results. The experiments using the SURE platform are valuable to evaluate how resilient is the sensor selection approach for traffic forecasting under different conditions.

1) System Model: We first introduce our sensor placement, prediction, and attacker models, and then formulate the problem of resilient sensor location selection.

a) Sensor and prediction model: We assume that a set $\mathcal{V}$ of possible sensor locations is given, and a designer can place at most N sensors at a set $\mathcal{S} \subset \mathcal{V}$ of locations. The designer uses the observations of the deployed sensors to predict a value using Gaussian process based regression.[1] For example, traffic measurements obtained from induction-loop sensors can be used to predict traffic situation at unobserved locations or in a future time. Given observed values $\mathrm{x}_{\mathcal{S}}$ at set $\mathcal{S}$, the predicted value is a random variable—called the predictor variable—Y, which follows a Gaussian distribution $\mathcal{N}\left(\mu_{\mathrm{Y}|\mathcal{S}}, \sigma^2_{\mathrm{Y}|\mathcal{S}}\right)$ with

$$\mu_{\mathrm{Y}|\mathcal{S}} = \mu_{\mathrm{Y}} + \sum_{Y\mathcal{S}} \sum_{\mathcal{S}\mathcal{S}}^{-1} (\mathrm{x}_{\mathcal{S}} - \mu_{\mathcal{S}}) \tag{3}$$

$$\sigma^2_{\mathrm{Y}|\mathcal{S}} = \sigma^2_{\mathrm{Y}} - \sum_{Y\mathcal{S}} \sum_{\mathcal{S}\mathcal{S}}^{-1} \sum_{\mathcal{S}Y} \tag{4}$$

where Σ is the (prior) covariance matrix of all the variables, while μ_{Y} and $\mu_{\mathcal{S}}$ are the (prior) means of the variables. These prior values are obtained using the SURE platform by running a large number of simulations, recording what sensors would observe at each location $\mathcal{V}$, and then computing the mean and covariance values from these observations.

b) Attacker model: Next, we introduce our model of DoS attacks against sensors. We assume that the attacker is resource bounded, so it can remove at most K of the sensors deployed by the designer. In practice, removing a sensor can model all forms of DoS type attacks, such as physical destruction, wireless jamming, or battery exhaustion. We also assume that the attacker is malicious in the sense that it will select a set of sensors to remove that will minimize the accuracy of prediction.

c) Problem formulation: We quantify the accuracy of predicting Y using the posterior variance $\sigma^2_{\mathrm{Y}|\mathcal{S}}$ (i.e., the lower the variance, the more accurate the prediction is). Then, the resilient sensor location selection problem can be formulated as an attacker–defender game defined as

$$\operatorname*{argmin}_{\mathcal{S}\subseteq\mathcal{V}:|\mathcal{S}|=N} \left(\max\nolimits_{\mathcal{A}\subseteq\mathcal{S}:|\mathcal{A}|=K} \sigma^2_{Y|(\mathcal{S}\setminus\mathcal{A})}\right). \tag{5}$$

Note that the designer (traffic engineer) first selects a set of sensor locations to minimize variance, and then the attacker removes a set of sensors to maximize the variance.

2) Heuristic Algorithm: The SURE platform allows blue and red teams to play the game and evaluate the resilience of sensor selection under arbitrary strategies. In the following, we focus on strategies that optimize the attacker and defender utilities respectively. The resilient sensor location selection problem is NP-hard [71], so we cannot hope to solve it in polynomial time. Consequently, we introduce an efficient heuristic algorithm (see Algorithm 2) for finding near-optimal selection in practice.

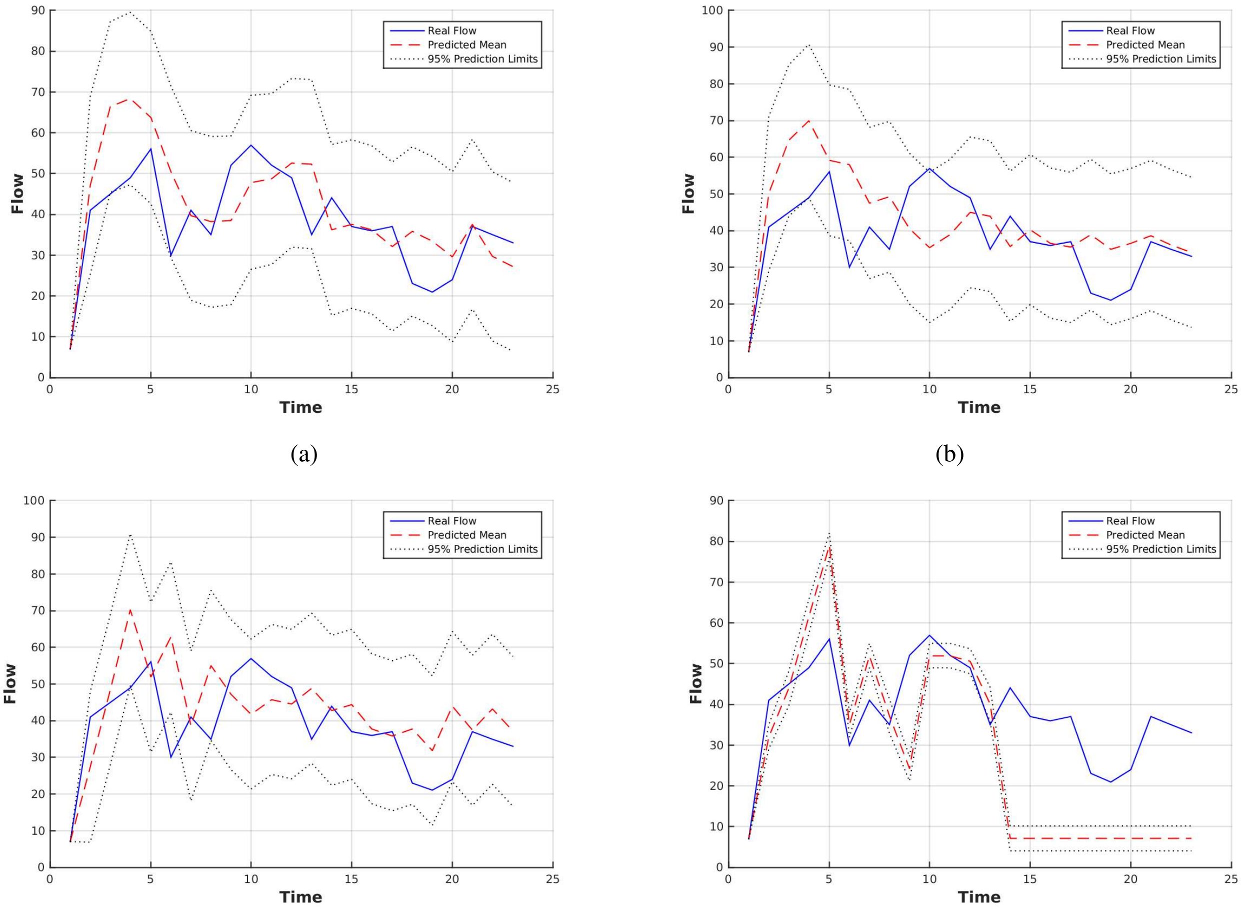
[1] Note that our approach can be generalized to predicting multiple values in a straightforward way, but we limit our discussion to a single value for ease of presentation.

Algorithm 2 Polynomial-Time Heuristic Algorithm for Resilient Sensor Location Selection

```
$\mathcal{S} \leftarrow \emptyset$
while $|\mathcal{S}| < K+1$ do
    $X^* \in \operatorname{argmin}_{X \in (\mathcal{V} \setminus \mathcal{S})} \sigma^2_{Y|X}$
    $\mathcal{S} \leftarrow \mathcal{S} \cup \{X^*\}$
end while
while $|\mathcal{S}| < N$ do
    $X^* \in \operatorname{argmin}_{X \in (\mathcal{V} \setminus \mathcal{S})} \max_{\mathcal{A} \in (\mathcal{S} \cup \{X\}) : |\mathcal{A}| = K} \sigma^2_{Y|((\mathcal{S} \cup \{X\}) \setminus \mathcal{A})}$
    $\mathcal{S} \leftarrow \mathcal{S} \cup \{X^*\}$
end while
return
$\mathcal{S}$
```

3) Simulation Results: We place traffic flow sensors around the Vanderbilt University campus in our simulation testbed, and evaluate the accuracy of a traffic predictor under various attack scenarios. First, we place eleven sensors at various locations, and we train a Gaussian process model for predicting traffic flow at another, unobserved location. Then, we let an attacker disable some of these sensors, we simulate the traffic flow, and plot the real and predicted traffic flow values, as well as the 95% prediction limits.

Fig. 17 shows the real (solid blue line) and predicted traffic values (dashed red line), as well as the 95% prediction limits (dotted line) as functions of time in various attack scenarios. First, Fig. 17(a) shows a baseline scenario without an attack. In this case, all sensors are working correctly, and the root-mean-square error (i.e., variance) of the prediction is only 10.54. Fig. 17(b) and (c) shows attacks increasing in size, disabling six and nine sensors, respectively. These results show that the predictor can withstand even attacks that disable a majority of the sensors, as the prediction error values remain low, 12.59 and 12.67, respectively. Finally, Fig. 17(d) shows a devastating attack that disables all but one sensors. The plot shows that this attack can effectively cripple the predictor, which is confirmed by the significantly higher prediction error value of 20.01. These figures are shown to the SURE user using the web-based interface allowing easy and effective analysis.



(a)

(b)

(c)

(d)

Fig. 17. ***Traffic flow prediction with various attack sizes: (a) Without attack; (b) 6 Sensors disabled; (c) 9 Sensors disabled; (d) 10 Sensors disabled.***

C. Resilient Traffic Signal Control

Management of traffic lights in urban transportation networks is a major challenge in modern traffic systems [72]. A common goal is to minimize congestion. Much prior work has now demonstrated that allowing for dynamic real-time control (as compared to fixed-time control) can significantly improve performance of optimized traffic light controllers [73]–[75]. A number of methods to perform optimization of closed-loop control systems have been proposed, where sensor measurements are used to dynamically adjust the timing of traffic light green-red cycles [76]–[79].

Although adaptive, state-aware strategies can offer tremendous gains in traffic control efficiency, they expose an attack surface that can be exploited to substantially increase congestion. For example, a common kind of adaptive control logic utilizes vehicle queue lengths in each direction, with light switching between red and green as a function of relative queue lengths. While such feedback-based switching can significantly increase efficiency, it also exposes a vulnerability of controllers to attacks on sensors from which queue length information is derived. An additional consideration which is crucial in modern transportation networks is that traffic lights on the network are often managed by multiple actors (e.g., municipalities).

We demonstrate how the SURE platform can be successfully used to systematically and efficiently explore these challenges in multi-intersection closed-loop traffic light control, where 1) traffic light controllers take into account relative queue lengths to determine red-green state of the traffic lights at an intersection; 2) controllers for all lights must be designed to work jointly so as to optimize overall traffic network performance; 3) sensors feeding data into the controllers are vulnerable to DoS attacks; and 4) intersections can be partitioned among a set of players, with own goals pertaining to congestion within their local municipal region, which are in general misaligned with global interests of the entire traffic network. Details of the theoretical approach can be found in [80].

To fully analyze this model using simulations, we need to explore varying configurations, such as alternative attack models and traffic patterns. In addition, it is crucial to explore these problems with a human in the loop to enable interactive configuration and vulnerability analysis. The SURE platform proves an ideal framework for such analysis.

1) Traffic Network and Controller Model: We introduce the control logic used in the paper and define the metrics to measure the efficiency of a transportation system.

Formally, a feedback traffic light controller has a predefined phase sequence $(p_0,\ldots,p_n)$. For each phase p_i, m_i is the minimum interval, M_i is the maximal interval, q_i is the average queue length of the lanes related to the ith phase, and θ_i is the threshold on the queue length of lanes blocked in the ith phase. The control logic is depicted in Algorithm 3 where t denotes the current time. The controller parameters we need to tune are $\Theta = (\Theta_0,\ldots,\Theta_m)$ where $\Theta_i = (\theta_0,\ldots\theta_{n_i})$ are the thresholds of the ith intersection. The global objective is to maximize average speed, $s(\Theta)$, over the entire traffic network.

Algorithm 3 Feedback Controller

```
Current Phase $P := p_0$, $t' := t$, $i := 0$.
loop
    $i_{next} := (i+1) \mod n$
    if $t - t' > m_i$ then
        if Reach to the maximum interval, $t - t' = M_i$ then
            Switch phase, $P = p_{i_{next}}$, $i = i_{next}$
        else if Find the congestion, $q_i < \theta_i$, $q_{i_{next}} \geq \theta_{i_{next}}$ then
            Switch phase, $P = p_{i_{next}}$, $i = i_{next}$
        end if
    end if
end loop
```

2) Game-Theoretic Approach to Resilient Closed-Loop Control: We investigate the consequences of DoS attacks on sensors, as well as the associated problem of resilient traffic signal control, whereby parameters of controllers are designed so as to be maximally resilient to such attacks. We formally model this as a Stackelberg game in which the leader is the traffic engineer who chooses controller parameters for the entire network, and the follower is an attacker who chooses a subset of K sensors to attack. We let A denote the set of attacker strategies, and assume that the attacker's goal is to minimize average speed over the entire network. Let $s(\Theta, a)$ denote the average speed obtained when the traffic controller parameters are Θ and attack a is deployed (which chooses the identities of K sensors to disable). A Stackelberg equilibrium of this game is a combination of strategies $(\Theta, a(\Theta))$ for the leader and the follower, respecively, such $a(\Theta') \in \arg\min_{a \in A} s(\Theta', a)$ for all Θ', and $\Theta \in \arg\max_t s(t, a(t))$.

In general, computing a Stackelberg equilibrium is computationally intractable in this setting. We therefore do so approximately. First, we compute for any controller configurations Θ an approximately optimal attack $a(\Theta)$ by greedily choosing K sensors to attack in order of marginal impact on average speed. Second, we optimize controller parameters greedily by optimizing one parameter at a time, computing an approximate attack a for each prospective parameter vector, until a local optimum is reached.

We evaluate this approach using the Vanderbilt University campus network with five selected intersections. The results are shown in Fig. 18. We can make two important observations: 1) controller parameters which are jointly optimized can result in a significant increase in average speed; and 2) explicitly building resilience into a controller via a game-theoretic approach above can dramatically improve its resilience to attacks, while maintaining high-quality performance when no attacks are present.

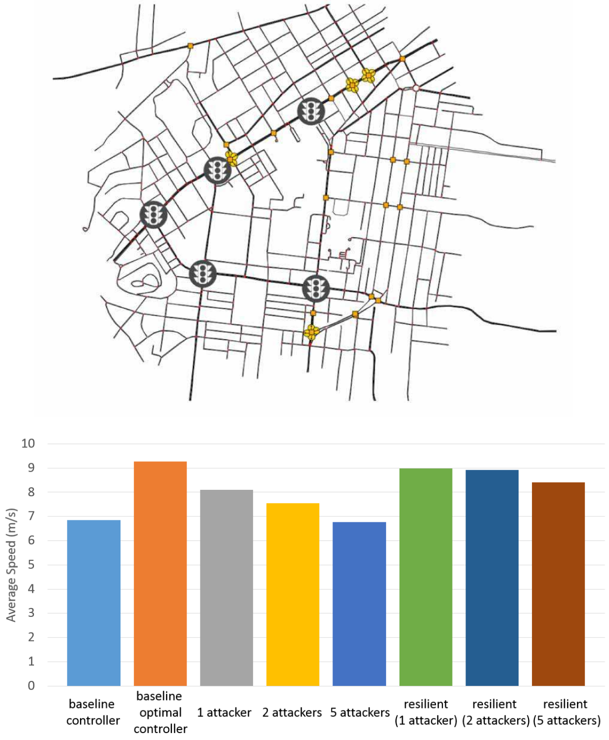


Fig. 18. ***Centralized control scenario. (Top) Map with the five optimized intersections indicated. (Bottom) Performance of optimized and baseline controllers, with and without attacks.***

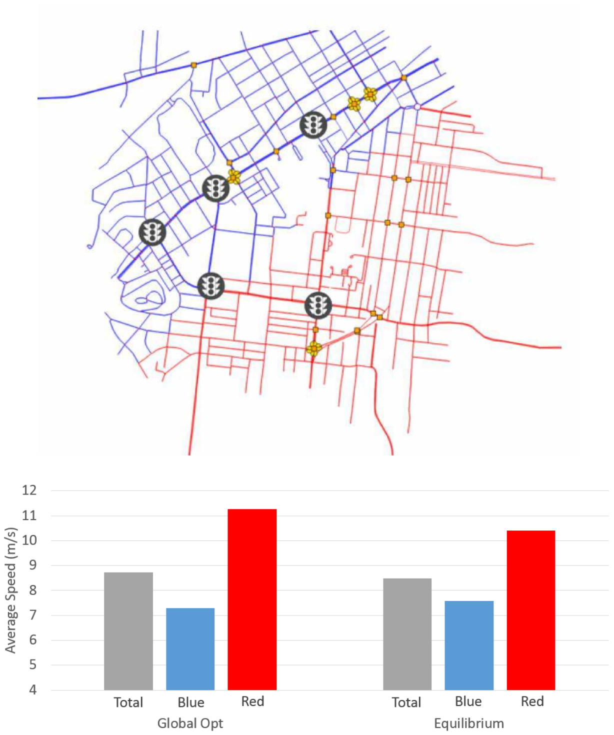


Fig. 19. ***Decentralized control scenario. (Top) Map with the five optimized intersections indicated in two regions of Vanderbilt campus. (Bottom) Performance of centralized and decentralized equilibrium controllers.***

3) Decentralized Traffic Signal Control: Transportation networks are large-scale interconnected systems, which typically cross municipal boundaries. An important challenge is to recognize that each municipal entity can, in principle, determine traffic network parameters toward its own specific ends. It is crucial to consider the impact of such decentralized decisions on overall traffic, as well as its resilience. To this end, we view the traffic network as partitioned among a set of players P, where each player $p \in P$ engineers the traffic light controllers on their part of the network. We denote the subset of parameters controlled by a player p as Θ_p, so that the full array of controller parameters is $(\Theta_1, \ldots, \Theta_{|P|})$. Each player is concerned with congestion, measured by average speed, only over their own portion of the network. Thus, we let $s_p(\Theta)$ denote average speed for the subnetwork controlled by a player p. Note that since $s(\Theta)$ is the overall average speed on the entire network, $s(\Theta) = \sum_p \alpha_p s_p(\Theta)$ for some $\alpha_p \geq 0$ with $\sum_p \alpha_p = 1$.

In this setting, we are concerned with Nash equilibrium controller policies, formally defined as a vector of controller parameters Θ such that $s_p(\Theta_p, \Theta_{-p}) \geq s_p(\Theta_{p'}, \Theta_{-p})$ for all p and alternative controller policies $\Theta_{p'}$ over player p's portion of the network. While computing such an equilibrium is in general intractable, we can approximately do so using best response dynamics, whereby a single player optimizes its control parameters one at a time, while the decisions of all others are fixed, until a fixed point of the process is reached.

Fig. 19 illustrates this model and approach by considering the Vanderbilt University campus network partitioned into two regions (red and blue). The controllers in each region are optimized only with respect to the average speed on the region in which they reside; however, since controller settings may impact traffic in another region, such interaction induces equilibrium controller parameter settings, each optimal given the other. A key observation is that in this case the overall network average speed is not significantly affected by the decentralized nature of the problem, even though equilibrium speed in the red region drops, while it increases in the blue region. We also considered a variation with the same map partitioned among three self-interested control regions, and found there as well that while differences are substantial between equilibrium and optimal controllers region by region, average speed over the entire network is not much affected by decentralization.

VII. CONCLUSION

Theoretical foundations and empirical research for resilience of CPS are extremely important since resilience can increase the adversary's level of effort required to achieve malicious objectives. Resilient CPS design can lead to systems that are highly resistant to malicious activities and can prevent large disruptions.

Our objective is to evaluate resilience of CPS in the presence of cyber attacks. The evaluation is based on attacker–defender games using simulations of sufficient fidelity. To rapidly synthesize complex heterogeneous simulations, we have developed a modeling integration framework and a tool infrastructure for attack and system modeling. The models are used by a model-based integration framework for heterogeneous and distributed simulations to support rapid design, synthesis, and evaluation of experiments. To achieve these goals, we have developed the SURE platform and demonstrated the approach for CPS transportation systems where SURE provides the necessary domain-specific languages, models, model translation and simulation driver tools to establish a coherent experimentation framework.

The SURE platform enables in-depth experimental evaluation of security and resilience that is necessary for developing the scientific foundations and technology of CPS. Theoretical analysis is accompanied by large amounts of experimental work and empirical observations use realistic CPS models and integrated simulations of tightly coupled cyber and physical components. Additionally, the platform allows the design and execution of controlled experiments of large-scale CPS by configuring the system and attack models. The platform enables measurement of adversary level of effort and potential gain, and therefore, such methods can be used for CPS engineering that aims at minimizing potential damage of cyber attacks. ■

REFERENCES

[1] R. Baheti and H. Gill, "Cyber-physical systems," *Impact Control Technol.*, vol. 12, pp. 161–166, Mar. 2011.

[2] J. Sztipanovits *et al.*, "Toward a science of cyber–physical system integration," *Proc. IEEE*, vol. 100, no. 1, pp. 29–44, Jan. 2012.

[3] H. Neema *et al.*, "SURE: An experimentation and evaluation testbed for CPS security and resilience: Demo abstract," in *Proc. 7th Int. Conf. Cyber-Phys. Syst. (ICCPS)*, 2016, Art. no. 27.

[4] R. Langner, "Stuxnet: Dissecting a cyberwarfare weapon," *IEEE Security Privacy*, vol. 9, no. 3, pp. 49–51, May 2011.

[5] J. Slay and M. Miller, "Lessons learned from the Maroochy water breach," in *Critical Infrastructure Protection*. 2007, pp. 73–82.

[6] M. K. Yoon, B. Liu, N. Hovakimyan, and L. Sha, "VirtualDrone: Virtual sensing, actuation, and communication for attack-resilient unmanned aerial systems," in *Proc. 8th ACM/IEEE Int. Conf. Cyber-Phys. Syst.*, Apr. 2017, pp. 143–154.

[7] K. Koscher *et al.*, "Experimental security analysis of a modern automobile," in *Proc. IEEE Symp. Secur. Privacy (SP)*, May 2010, pp. 447–462.

[8] S. Checkoway *et al.*, "Comprehensive experimental analyses of automotive attack surfaces," in *Proc. USENIX Secur. Symp.*, San Francisco, CA, USA, 2011, pp. 1–6.

[9] J. Reilly, S. Martin, M. Payer, and A. M. Bayen, "On cybersecurity of freeway control systems: Analysis of coordinated ramp metering attacks," in *Proc. Transp. Res. Board 94th Annu. Meet.*, 2015, paper 15–5248.

[10] J. P. Hubaux, S. Capkun, and J. Luo, "The security and privacy of smart vehicles," *IEEE Security Privacy*, vol. 2, no. 3, pp. 49–55, May/Jun. 2004.

[11] D. Halperin, T. S. Heydt-Benjamin, K. Fu, T. Kohno, and W. H. Maisel, "Security and privacy for implantable medical devices," *IEEE Perv. Comput.*, vol. 7, no. 1, pp. 30–39, Jan. 2008.

[12] Y. Mo *et al.*, "Cyber–physical security of a smart grid infrastructure," *Proc. IEEE*, vol. 100, no. 1, pp. 195–209, Jan. 2012.

[13] M. Cheminod, L. Durante, and A. Valenzano, "Review of security issues in industrial networks," *IEEE Trans. Ind. Informat.*, vol. 9, no. 1, pp. 277–293, Feb. 2013.

[14] B. Hoh, M. Gruteser, H. Xiong, and A. Alrabady, "Enhancing security and privacy in traffic-monitoring systems," *IEEE Perv. Comput.*, vol. 5, no. 4, pp. 38–46, Oct. 2006.

[15] S. Amin, G. A. Schwartz, and A. Hussain, "In quest of benchmarking security risks to cyber-physical systems," *IEEE Network*, vol. 27, no. 1, pp. 19–24, Jan. 2013.

[16] A. A. Cardenas, S. Amin, and S. Sastry, "Secure control: Towards survivable cyber-physical systems," in *Proc. 28th Int. Conf. Distrib. Comput. Syst. Workshops*, Jun. 2008, pp. 495–500.

[17] J. Giraldo, E. Sarkar, A. A. Cardenas, M. Maniatakos, and M. Kantarcioglu, "Security and privacy in cyber-physical systems: A survey of surveys," *IEEE Des. Test.*, vol. 34, no. 4, pp. 7–17, Aug. 2017.

[18] G. Martins, A. Moondra, A. Dubey, A. Bhattacharjee, and X. D. Koutsoukos, "Computation and communication evaluation of an authentication mechanism for time-triggered networked control systems," *Sensors*, vol. 16, no. 8, p. 1166, 2016.

[19] L. Pike, J. Sharp, M. Tullsen, P. C. Hickey, and J. Bielman, "Secure automotive software: The next steps," *IEEE Software*, vol. 34, no. 3, pp. 49–55, May/Jun. 2017.

[20] A. Banerjee, K. K. Venkatasubramanian, T. Mukherjee, and S. K. S. Gupta, "Ensuring safety, security, and sustainability of mission-critical cyber–physical systems," *Proc. IEEE*, vol. 100, no. 1, pp. 283–299, Jan. 2012.

[21] M. W. Whalen, D. Cofer, and A. Gacek, "Requirements and architectures for secure vehicles," *IEEE Software*, vol. 33, no. 4, pp. 22–25, Jul./Aug. 2016.

[22] A. A. Cárdenas, S. Amin, Z.-S. Lin, Y.-L. Huang, C.-Y. Huang, and S. Sastry, "Attacks against process control systems: Risk assessment, detection, and response," in *Proc. 6th ACM Symp. Inf., Comput. Commun. Secur.*, Mar. 2011, pp. 355–366.

[23] F. Pasqualetti, F. Dörfler, and F. Bullo, "Attack detection and identification in cyber-physical systems," *IEEE Trans. Autom. Control*, vol. 58, no. 11, pp. 2715–2729, Nov. 2013.

[24] A. Ghafouri, W. Abbas, A. Laszka, Y. Vorobeychik, and X. Koutsoukos, "Optimal thresholds for anomaly-based intrusion detection in dynamical environments," in *Proc. Int. Conf. Decision Game Theory Secur.*, 2016, pp. 415–434.

[25] D. I. Urbina *et al.*, "Limiting the impact of stealthy attacks on industrial control systems," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.*, Oct. 2016, pp. 1092–1105.

[26] Q. Zhu and T. Başar, "Robust and resilient control design for cyber-physical systems with an application to power systems," in *Proc. 50th IEEE Conf. Decision Control Eur. Control Conf. (CDC-ECC)*, Dec. 2011, pp. 4066–4071.

[27] M. Zhu and S. Martínez, "Attack-resilient distributed formation control via online adaptation," in *Proc. 50th IEEE Conf. Decision Control Eur. Control Conf. (CDC-ECC)*, Dec. 2011, pp. 6624–6629.

[28] H. J. LeBlanc, H. Zhang, X. Koutsoukos, and S. Sundaram, "Resilient asymptotic consensus in robust networks," *IEEE J. Sel. Areas Commun.*, vol. 31, no. 4, pp. 766–781, Apr. 2013.

[29] Y. Shoukry *et al.*, "SMT-based observer design for cyber-physical systems under sensor attacks," in *Proc. 7th ACM/IEEE Int. Conf. Cyber-Phys. Syst. (ICCPS)*, Apr. 2016, pp. 1–10.

[30] 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 *Proc. 2nd ACM Int. Conf. High Confidence Netw. Syst.*, Apr. 2013, pp. 65–74.

[31] M. Pajic, J. Weimer, N. Bezzo, O. Sokolsky, G. J. Pappas, and I. Lee, "Design and implementation of attack-resilient cyberphysical systems: With a focus on attack-resilient state estimators," *IEEE Control Syst.*, vol. 37, no. 2, pp. 66–81, Apr. 2017.

[32] C. M. Davis, J. E. Tate, H. Okhravi, C. Grier, T. J. Overbye, and D. Nicol, "SCADA cyber security testbed development," in *Proc. 38th North Amer. Power Symp. (NAPS)*, Sep. 2006, pp. 483–488.

[33] T. Kropp, "System threats and vulnerabilities [power system protection]," *IEEE Power Energy Mag.*, vol. 4, no. 2, pp. 46–50, Mar. 2006.

[34] M. Mallouhi, Y. Al-Nashif, D. Cox, T. Chadaga, and S. Hariri, "A testbed for analyzing security of SCADA control systems (TASSCS)," in *Proc. IEEE PES Innov. Smart Grid Technol. (ISGT)*, Jan. 2011, pp. 1–7.

[35] S. Sridhar, A. Hahn, and M. Govindarasu, "Cyber–physical system security for the electric power grid," *Proc. IEEE*, vol. 100, no. 1, pp. 210–224, Jan. 2012.

[36] A. Ashok, P. Wang, M. Brown, and M. Govindarasu, "Experimental evaluation of cyber attacks on automatic generation control using a CPS security testbed," in *Proc. IEEE Power Energy Soc. Gen. Meet.*, Jul. 2015, pp. 1–5.

[37] B. Ghena, W. Beyer, A. Hillaker, J. Pevarnek, and J. A. Halderman, "Green lights forever: Analyzing the security of traffic infrastructure," in *Proc. 8th USENIX Workshop Offensive Technol. (WOOT)*, 2014, pp. 1–10.

[38] M. Maroti, R. Kereskényi, T. Kecskés, P. Völgyesi, and A. Lédeczi, "Online collaborative environment for designing complex computational systems," in *Proc. Int. Conf. Comput. Sci. (ICCS)*, 2014.

[39] G. Hemingway, H. Neema, H. Nine, J. Sztipanovits, and G. Karsai, "Rapid synthesis of high-level architecture-based heterogeneous simulation: A model-based integration approach," *Simulation*, vol. 88, no. 2, pp. 217–232, 2012.

[40] *IEEE Standard for Modeling and Simulation (M&S) High Level Architecture (HLA)—Framework and Rules*, IEEE Standard 15162010, 2010, pp. 1–38.

[41] A. Varga, "The OMNeT++ discrete event simulation system," in *Proc. Eur. Simulation Multiconf. (ESM)*, Prague, Czech Republic, 2001.

[42] D. Krajzewicz, J. Erdmann, M. Behrisch, and L. Bieker, "Recent development and applications of SUMO—Simulation of Urban MObility," *Int. J. Adv. Syst. Meas.*, vol. 5, nos. 3–4, pp. 128–138, Dec. 2012.

[43] Matlab/Simulink. [Online]. Available: https://www.mathworks.com/products/simulink.html

[44] A. Lédeczi *et al.*, "Composing domain-specific design environments," *IEEE Comput.*, vol. 34, no. 11, pp. 44–51, Nov. 2001.

[45] E. Long, A. Misra, and J. Sztipanovits, "Increasing productivity at Saturn," *Computer*, vol. 31, no. 8, pp. 35–43, Aug. 1998.

[46] J. L. Mathe *et al.*, "A model-integrated, guideline-driven, clinical decision-support system," *IEEE Software*, vol. 26, no. 4, pp. 54–61, Jul. 2009.

[47] S. Shetty, S. Neema, and T. Bapty, "Model based self adaptive behavior language for large scale real time embedded systems," in *Proc. 11th IEEE Int. Conf. Workshop Eng. Comput.-Based Syst.*, May 2004, pp. 478–483.

[48] H. Bagheri and K. Sullivan, "Monarch: Model-based development of software architectures," in *Model Driven Engineering Languages and Systems*. 2010, pp. 376–390.

[49] J. Bézivin, C. Brunette, R. Chevrel, F. Jouault, and I. Kurtev, "Bridging the generic modeling environment (GME) and the eclipse modeling framework (EMF)," in *Proc. Best Pract. Model Driven Softw. Develop. (OOPSLA)*, vol. 5. 2005.

[50] P. Bunus, "A simulation and decision framework for selection of numerical solvers in scientific computing," in *Proc. 39th Annu. Symp. Simulation*, Washington, DC, USA, Apr. 2006, pp. 178–187.

[51] J. A. Stankovic *et al.*, "VEST: An aspect-based composition tool for real-time systems," in *Proc. 9th IEEE Real-Time Embedded Technol. Appl. Symp.*, May 2003, pp. 58–69.

[52] S. Chacon and B. Straub, *Pro Git*, 2nd ed. Berkely, CA, USA: Apress, 2014.

[53] *Amazon Simple Storage Service (Amazon S3)*. [Online]. Available: https://aws.amazon.com/s3/

[54] (Apr. 2017). *Openstreetmap*. [Online]. Available: http://openstreetmap. org

[55] " The JSON data interchange format," ECMA, Geneva, Switzerland, Tech. Rep. Standard ECMA-404 1st ed., Oct. 2013. [Online]. Available: http://www.ecma-international.org/publications/files/ECMA-ST/ECMA-404.pdf

[56] H. Neema *et al.*, "Model-based integration platform for FMI co-simulation and heterogeneous simulations of cyber-physical systems," in *Proc. 10 th Int. Modelica Conf.*, Lund, Sweden, 2014, no. 096, pp. 235–245.

[57] Portico. (Dec. 2016). [Online]. Available: https://github.com/openlvc/portico

[58] K. Jensen, L. M. Kristensen, and L. Wells, "Coloured Petri nets and CPN tools for modelling and validation of concurrent systems," *Int. J. Softw. Tools Technol. Transfer*, vol. 9, nos. 3–4, pp. 213–254, 2007.

[59] M. Behrisch, L. Bieker, J. Erdmann, and D. Krajzewicz, "SUMO—Simulation of Urban MObility: An overview," in *Proc. 3rd Int. Conf. Adv. Syst. Simulation (SIMUL)*, 2011, pp. 63–68.

[60] *Traindirector*. [Online]. Available: http://www.backerstreet.com/traindir/

[61] D. P. Chassin, J. C. Fuller, and N. Djilali, "GridLAB-D: An agent-based simulation framework for smart grids," *J. Appl. Math.*, vol. 2014, 2014, Art. no. 492320.

[62] *Inet Framework*. [Online]. Available: http://inet.omnetpp.org/

[63] C. F. Daganzo, "The cell transmission model: A dynamic representation of highway traffic consistent with the hydrodynamic theory," *Transp. Res. B, Methodol.*, vol. 28, no. 4, pp. 269–287, Aug. 1994.

[64] A. Laszka, B. Potteiger, Y. Vorobeychik, S. Amin, and X. Koutsoukos, "Vulnerability of transportation networks to traffic-signal tampering," in *Proc. 7th ACM/IEEE Int. Conf. Cyber-Phys. Syst. (ICCPS)*, Apr. 2016, pp. 1–10.

[65] C. E. Rasmussen and C. K. I. Williams, *Gaussian Processes for Machine Learning* (Adaptive Computation and Machine Learning). Cambridge, MA, USA: MIT Press, 2005.

[66] Y. Xie, K. Zhao, Y. Sun, and D. Chen, "Gaussian processes for short-term traffic volume forecasting," *Transp. Res. Rec., J. Transp. Res. Board*, vol. 2165, no. 1, pp. 69–78, 2010.

[67] J. Chen *et al.*, "Decentralized data fusion and active sensing with mobile sensors for modeling and predicting spatiotemporal traffic phenomena," in *Proc. 28th Conf. Uncertainty Artif. Intell. (UAI)*, 2012, pp. 163–173.

[68] P. Kou, F. Gao, and X. Guan, "Sparse online warped Gaussian process for wind power probabilistic forecasting," *Appl. Energy*, vol. 108, pp. 410–428, Aug. 2013.

[69] Y. Bazi, N. Alajlan, and F. Melgani, "Improved estimation of water chlorophyll concentration with semisupervised Gaussian process regression," *IEEE Trans. Geosci. Remote Sens.*, vol. 50, no. 7, pp. 2733–2743, Jul. 2012.

[70] I. Nevat, G. W. Peters, and I. B. Collings, "Location-aware cooperative spectrum sensing via Gaussian processes," in *Proc. 13th Austral. Commun. Theory Work. (AusCTW)*, Jan. 2012, pp. 19–24.

[71] A. Laszka, Y. Vorobeychik, and X. Koutsoukos, "Resilient observation selection in adversarial settings," in *Proc. IEEE 54th Annu. Conf. Decision Control (CDC)*, Dec. 2015, pp. 7416–7421.

[72] D. Helbing, J. Siegmeier, and S. Lämmer, "Self-organized network flows," in *Proc. NHM*, 2007, vol. 2. no. 2, pp. 193–210.

[73] M. E. Fouladvand, M. R. Shaebani, and Z. Sadjadi, "Intelligent controlling simulation of traffic flow in a small city network," *J. Phys. Soc. Jpn*, vol. 73, no. 11, pp. 3209–3214, 2004.

[74] C. Gershenson and D. A. Rosenblueth, "Self-organizing traffic lights at multiple-street intersections," *Complexity*, vol. 17, no. 4, pp. 23–39, 2012.

[75] S. Lämmer, H. Kori, K. Peters, and D. Helbing, "Decentralised control of material or traffic flows in networks using phase-synchronisation," *Phys. A, Stat. Mech. Appl.*, vol. 363, no. 1, pp. 39–47, Apr. 2006.

[76] M. Zhong, S. Sharma, and P. Lingras, "Genetically-designed time delay neural networks for multiple-interval urban freeway traffic flow forecasting," *Neural Inf. Process. Lett. Rev.*, vol. 10, nos. 8–9, pp. 201–209, 2006.

[77] S. Mikami and Y. Kakazu, "Genetic reinforcement learning for cooperative traffic signal control," in *Proc. 1st IEEE Conf. Evol. Comput., IEEE World Congr. Comput. Intell.*, vol. 1. Jun. 1994, pp. 223–228.

[78] T. Royani, J. Haddadnia, and M. Alipoor, "Traffic signal control for isolated intersections based on fuzzy neural network and genetic algorithm," in *Proc. 10th WSEAS Int. Conf. Signal Process., Comput. Geometry Artif. Vis.*, 2010, pp. 87–91.

[79] R. Hoar, J. Penner, and C. Jacob, "Evolutionary swarm traffic: If ant roads had traffic lights," in *Proc. Congr. Evol. Comput. (CEC)*, vol. 2. May 2002, pp. 1910–1915.

[80] J. Lou and Y. Vorobeychik, "Decentralization and security in dynamic traffic light control," in *Proc. Symp. Bootcamp Sci. Secur.*, 2016, pp. 90–92.

ABOUT THE AUTHORS

Xenofon Koutsoukos (Senior Member, IEEE) received the Ph.D. degree in electrical engineering from the University of Notre Dame, Notre Dame, IN, USA, in 2000.

He is a Professor with the Department of Electrical Engineering and Computer Science and a Senior Research Scientist with the Institute for Software Integrated Systems (ISIS), Vanderbilt University, Nashville, TN, USA. Previously, he was a Member of Research Staff at the Xerox Palo Alto Research Center (PARC) (2000–2002), working in the embedded collaborative computing area. He has published numerous journal and conference papers and he is coinventor of four U.S. patents. His research work is in the area of cyber–physical systems with emphasis on formal methods, distributed algorithms, security and resilience, diagnosis and fault tolerance, and adaptive resource management.

Prof. Koutsoukos was the recipient of the National Science Foundation (NSF) Career Award in 2004, the Excellence in Teaching Award in 2009 from the Vanderbilt University School of Engineering, and the 2011 NASA Aeronautics Research Mission Directorate (ARMD) Associate Administrator (AA) Award in Technology and Innovation.

Gabor Karsai (Senior Member, IEEE) received the B.Sc., M.Sc., and Dr. Techn. degrees from the Technical University of Budapest, Budapest, Hungary, in 1982, 1984, and 1988, respectively, and the Ph.D. degree from Vanderbilt University, Nashville, TN, USA, in 1988.

He is Professor of Electrical Engineering and Computer Science at Vanderbilt University and Senior Research Scientist and Associate Director at the Institute for Software-Integrated Systems. He has over 25 years of experience in research on systems and software engineering. He conducts research in the design and implementation of embedded systems, in programming tools for visual programming environments, and in the theory and practice of model-integrated computing.

Aron Laszka received the Ph.D. degree in computer science *summa cum laude* from the Budapest University of Technology and Economics, Budapest, Hungary, in 2014.

In 2013, he was a Visiting Research Scholar at the Pennsylvania State University. Currently, he is a Research Assistant Professor in the Department of Electrical Engineering and Computer Science, Vanderbilt University, Nashville, TN, USA. His research interests broadly revolve around the security and resilience of cyber–physical systems, the economics of cybersecurity, and game-theoretic modeling of security problems. Previously, he was a Postdoctoral Scholar at the University of California Berkeley, Berkeley, CA, USA, from 2015 to 2016, and a Postdoctoral Research Scholar at Vanderbilt University from 2014 to 2015.

Himanshu Neema received the M.S. degree in computer science from Vanderbilt University, Nashville, TN, USA.

He is a System Architect at the Institute for Software Integrated Systems at Vanderbilt University. His primary research interests include modeling and simulation, model-integrated computing, distributed simulations, artificial intelligence, constraint programming, and planning and scheduling. He has 19+ years of experience in research and development of software applications covering these areas and has coauthored 20+ research publications.

Bradley Potteiger received the B.S. degree in computer engineering from the University of Maryland, Baltimore County, Baltimore, MD, USA and the M.S. degree in electrical engineering from Vanderbilt University, Nashville, TN, USA, where he is currently working toward the Ph.D. degree in the Department of Electrical Engineering and Computer Science, with a research affiliation at the Institute of Software Integrated Systems.

His research at Vanderbilt is focused on cyber–physical system (CPS) security for protection of safety-critical systems.

Peter Volgyesi is a Research Scientist at the Institute for Software Integrated Systems, Vanderbilt University, Nashville, TN, USA. He is one of the architects of the Generic Modeling Environment, a widely used metaprogrammable visual modeling tool, and WebGME, its modern web-based variant. As Principal Investigator on two National Foundation of Science (NSF)-funded projects, he and his team recently developed a low-power software-defined radio platform (MarmotE) and a component-based development toolchain targeting multicore SoC architectures for wireless cyber–physical systems. His research interests include wireless sensor networks, model-based engineering, and signal processing.

Yevgeniy Vorobeychik (Member, IEEE) received the B.S. degree in computer engineering from Northwestern University, Evanston, IL, USA and the M.S.E. and Ph.D. degrees in computer science and engineering from the University of Michigan, Ann Arbor, MI, USA, in 2004 and 2008, respectively.

He is an Assistant Professor of Computer Science, Computer Engineering, and Biomedical Informatics at Vanderbilt University, Nashville, TN, USA. Previously, he was a Principal Research Scientist at Sandia National Laboratories. Between 2008 and 2010, he was a Postdoctoral Research Associate at the Computer and Information Science Department, University of Pennsylvania, Philadelphia, PA, USA. His work focuses on game-theoretic modeling of security and privacy, adversarial machine learning, algorithmic and behavioral game theory and incentive design, optimization, agent-based modeling, complex systems, network science, and epidemic control.

Dr. Vorobeychik received a National Science Foundation (NSF) CAREER award in 2017, and was invited to give an IJCAI-16 early career spotlight talk. He was nominated for the 2008 ACM Doctoral Dissertation Award and received honorable mention for the 2008 IFAAMAS Distinguished Dissertation Award.

Janos Sztipanovits (Life Fellow, IEEE) graduated (*summa cum laude*) from the Technical University of Budapest, Budapest, Hungary, in 1970 and received the Ph.D. degree from the Hungarian Academy of Sciences, Budapest, Hungary, in 1980.

He is currently the E. Bronson Ingram Distinguished Professor of Engineering at Vanderbilt University, Nashville, TN, USA, and founding director of the Institute for Software Integrated Systems. His current research interest includes the foundation and applications of model and component-based design of cyber–physical systems, design space exploration and systems-security codesign technology.

Dr. Sztipanovits is an external member of the Hungarian Academy of Sciences. He was founding chair of the ACM Special Interest Group on Embedded Software (SIGBED).