Opportunities and Challenges in Monitoring Cyber-physical Systems Security

Borzoo Bonakdarpour[1], Jyotirmoy V. Deshmukh[2], and Miroslav Pajic[3]

[1] Iowa State University, USA, borzoo@iastate.edu
[2] University of Southern California, USA, jdeshmuk@usc.edu
[3] Duke University, USA, miroslav.pajic@duke.edu

Abstract. Technological advances in distributed *cyber-physical systems* (CPS) will fundamentally alter the way present and future human societies lead their lives. From a security or privacy perspective, a (multi-agent) cyber-physical system is a network of sensors, actuators, and computation nodes, i.e., a system with multiple attack surfaces and latent exploits that originate both through software attacks and physical attacks. In this paper, we argue that we are in pressing need to bring about a paradigm shift in software development for multi-agent CPS. To this end, security and privacy policies should be made a critical ingredient of agent interfaces with a goal of ensuring both localized safety and privacy for each agent, as well as guaranteeing global system safety and security. We present our vision on new theory, algorithms, and tools to foster a culture of secure-by-design multi-agent CPS.

1 Introduction

Human societies of tomorrow will be immersed in multi-agent cyber-physical systems (CPS). Examples include autonomous and semi-autonomous cars coupled with intelligent transportation systems as well as fleets of unmanned aerial vehicles (UAVs) performing mundane jobs like package delivery, and teams of rescue robots in disaster management scenarios. A key feature of these systems is that they consist of networked *multi-agent cyber* components that interact with the *physical* environment. Informally, a CPS is a system that combines a *plant*, i.e., a mechanical, electrical or hydraulic component that has temporal behavior which follows the laws of physics, controlled by an embedded software *controller*. A multi-agent CPS consists of two or more such CPSs with the ability to communicate with each other or with a central agent. It is tempting to think of a multi-agent CPS as just a larger CPS with several plants and controllers, but what distinguishes a multi-agent CPS from an ordinary CPS is the decoupling between individual agents. Often, agents in such a multi-agent CPS are autonomous, i.e., have some degree of freedom in controlling their behavior without the intervention from an external agent, or are semi-autonomous, i.e., they have the ability to switch control between a human operator and an embedded software controller.

In the past few years, we have seen the catastrophic levels of damage that attacks on cyber-physical systems can cause; examples include the blackout of

the Ukrainian power grid in 2015 [30], and the MIRAI botnet that made use of Internet of Things (IoT) devices to launch Distributed Denial-of-Service attacks [25]. Some types of cyber-induced attacks can have physical impacts; examples include several examples where automobile security was compromised [11,26,37], including a wireless hack on a Jeep vehicle in a controlled setting that received attention in popular media [18]. In the domain of aerial vehicles, examples include a GPS spoofing attack that allegedly led to the abduction of a US drone [38]. It is clear that the need for security and trust in cyberspace is fast changing into a need for secure and trustworthy *cyber-physical spaces*.

As a multi-agent CPS is a network of sensors, actuators, and computation nodes linked through communication channels, from a security perspective, such a system presents a plethora of attack surfaces. Direct attacks on such systems, as well as latent vulnerabilities can attract both software as well as physical attacks. Here, by software attacks we mean traditional cyber attacks that target communication of a CPS agent with its external world by seeking to compromise its availability, corrupt its data integrity, or lead to a loss of its data confidentiality. By physical attacks, we mean an adversarial action that can either learn the internal physical state of the system by observing its input/output behavior, alter its internal physical state by injecting commands or control actions, or use actual physical phenomena to induce unsafe behavior. Note that these categories are not mutually exclusive, and often attacks can be constructed by exploiting vulnerabilities in both the software and physical domains.

Our position is based on the premise that for a multi-agent CPS, there is a pressing need to design a framework that supports a diverse collection of security and privacy policies, but more importantly, supports reasoning about the impact of such policies on the safety of each agent in isolation, and also on the safety, security and privacy of agents at the level of the multi-agent CPS as a whole. More specifically, we argue that to achieve a paradigm shift in CPS security, we need to pursue the following objectives:

- The first step for systematic and formal reasoning about security or privacy is to have a *machine-checkable* language/logic that can express complex policies such as information flow in the context of CPS.
- This language/logic can then be used to monitor and enforce policies at the level of individual agents through careful design and implementation of sensor instrumentation at system level to gather the data required to evaluate the policies.
- Monitoring and enforcement of policies also needs to be done in a compositional fashion at the level of multiple agents in the CPS to reason about the impact of such policies at the level of the entire system.
- Finally, the language and monitoring/enforcement mechanisms need to be realized in real-world scenarios and systems with an eye on the next generation of CPSs that will play a crucial life in our daily lives.

We elaborate on our position on each of these objectives and our view on addressing them in Sections 2 – 4.

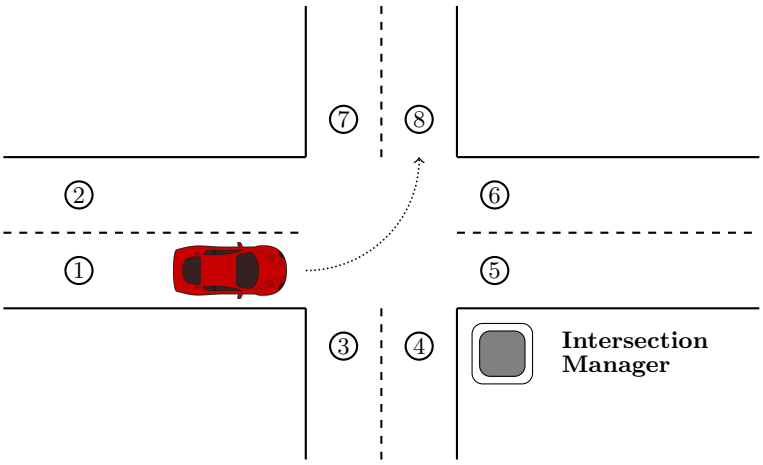

Fig. 1. Depiction of an autonomous intersection manager system.

2 Logic-based Expression of Security and Privacy Policies for CPS

Signal Temporal Logic (STL) is a machine-checkable logical formalism that was first introduced in the context of specifying properties of mixed-signal circuits [31]. There has been considerable interest in the use of STL for specifying industrial-scale embedded systems and an ecosystem of monitoring and test-generation tools has evolved around the logic [2, 3, 13, 15, 20, 21, 24]. We envision two extensions to STL to express security properties on confidentiality, integrity, and availability as well as temporal constraints that counter side-channel attacks.

STL extension for security. Our first proposed extension is *Security-Aware Signal Temporal Logic* (SA-STL), that introduces common security primitives as first-class predicates in the logic. This will allow designers to express security properties and constraints in a uniform, *machine-checkable* language. The key advantage of using SA-STL is that it inherits quantitative semantics of STL, which will allow us to quantify the *degree of security* of the system. SA-STL will also include security constraints that are stochastic in nature by allowing probabilistic predicates such as those allowed by Stochastic STL [29]. As this logic can reason over real-valued signals, it allows seamless reasoning over physical signals and quantities in a single logic. Consider for example the scenario shown in Fig. 1, where the car shown wishes to cross the intersection from the lane marked 1 to the one marked 8. The SA-STL formula (1) says that, "*if the car receives a message from the intersection manager granting permission to use the intersection, then, the car has a window of time in the future, where subject to the constraints imposed by the car's dynamics, the car can cross the intersection*

in a same fashion." We can express the following property in SA-STL as follows:

$$\mathbf{G}\left(\begin{pmatrix}\texttt{lane} = 1 \ \wedge \\ \texttt{recvEncMsg} = \texttt{granted} \wedge \\ \mathbf{F}_{[0,10]}\texttt{authSender} = \texttt{IntMgr}\end{pmatrix} \implies (\texttt{accel} < 2 \ \mathbf{U}_{[3,10]} \ \texttt{lane} = 8)\right) \quad (1)$$

We remark that this is just one aspect of the security policy that specifies the timeliness of crossing and authenticity of the received message. We observe that the above security policy can be expressed as a conjunction of separate parts that monitor the transmission of the request signal, reception of the grant signal, continuous monitoring of the physical signals corresponding to acceleration and position in its control unit, and transmission of successful intersection navigation once it reaches the desired lane.

Hyper logics. A large set of important information-flow security and privacy policies are inexpressible in trace-based variants of temporal logics (such as SA-STL). Although existing hyper logics such as HyperLTL [12] can express complex information flow policies, they currently do not allow explicit timing constraints and real-valued signals. Thus, we propose to design a new logic called *Hyper-MTL* that will allow explicit quantification over traces as well as timed temporal operators that enforce timing constraints across multiple traces. For example, we envision a timed until operator that enforces a time interval for the eventuality part of the operator as well as an error bound which allows events to happen within that bound but across multiple traces. For example, by formula $\varphi = \forall\pi.\forall\pi'.a_\pi \, \mathbf{U}_I^j \, b_{\pi'}$, we mean that in every pair of traces, b should occur within explicit time interval I, but occurrences of b in π and π' can take place in a sliding window such that occurrences are not j units apart. Thus, if $I = [0, \infty)$, then the sliding window can move at any point along the time. This will allow us to express protection policies against many side-channel timing attacks. This logic can then be extended to have predicates over real-valued signals, similar to STL, and we can formulate quantitative semantics to help obtain a notion of *robust* satisfaction.

As each agent in a multi-agent CPS is effectively a *hybrid dynamical system with inputs and outputs*, the above frameworks can employ recent research results on attack-resilient control and attack detection to synthesize observers that identify the conditions under which an agent is compromised. Thus, one can investigate mapping observers synthesized in this fashion into a SA-STL-based security or privacy policy expressed as a hyperproperty in the newly proposed hyper logic.

3 Monitor Synthesis and Resource-aware Monitoring Algorithms

Runtime monitoring is a technique commonly used for protecting a CPS against uncertainties in its environment. Runtime monitoring enables (1) automatic

identification of the minimal number of states or program variables to monitor so as to make monitoring *minimally intrusive*, (2) the use of quantitative, predictive monitoring that is resource-optimal and can prevent a security violation before it occurs, (3) the use of quantitative trust management [5, 42] to dynamically monitor trust levels of agents in the multi-agent CPS, and use trust as a mechanism to synthesize distributed observations through a multiplicity of agents and sensors. Our view is to design multi-faceted algorithms to monitor complex security and privacy policies in CPS on several fronts.

Robust and predictive monitoring. We advocate combining robust online monitoring for STL [14] and predictive monitoring for Metric Temporal Logic [16]. We believe that robust predictive monitoring can provide nuanced information about probabilistic and quantitative information of future security risks, allowing earlier preventive actions.

Robust monitoring of hyper logics. We envision algorithms for monitoring timed hyperproperties with real-valued signals. These algorithms will expand on previous efforts (e.g., [1, 6, 7, 9, 17]). Such a monitoring algorithm will take as input either (1) concurrent output traces of an instrumented running system, (2) offline logs of past executions, or (3) runtime traces and an abstract model of the system [8] as well as a set of formulas. The algorithms will evaluate the formulas and emit satisfaction/violation verdicts on the input online/offline traces. In case of violation, these verdicts will be used to take action on maintaining system safety or privacy. Following the recent trend, monitoring can be done on a GPU [4] or FPGA [19] device to minimize the impact of probe effects on the system under inspection and also achieve highly efficient resource management.

Sensor instrumentation. Sensor instrumentation for monitoring under architectural constraints is inevitable to achieve effective CPS monitoring. One approach is to exploit logging schemes for monitoring of distributed controller networks; for a single control loop (i.e., feature), [33, 34, 40, 41] introduce conditions that the instrumentation points need to satisfy to ensure full system observability with continuous monitoring, even in the presence of malicious components. Similarly, when intermittent monitoring is used, extensions of the techniques from [22, 23, 27, 28] can be utilized. This would effectively also allow for attack detection and identification of compromised components.

4 Compositional Runtime Enforcement of CPS Security Policies

Multi-agent CPS are inherently component-based. Thus, it is natural to think of decompositional methods that partition the overall system security, privacy or safety into assumptions and guarantees at the level of individual agents. There are two key building blocks that give us dynamic assurance of meeting security/privacy policies at run time: (1) each agent monitors the assumptions specified by the security/privacy policy of the system on the inputs it receives from

other agents or the environment, (2) a runtime enforcement system uses various techniques to enforce safe behavior of the CPS agent's actuation system, and to enforce the guarantees provided by the agent's outward communication to other agents. We discussed the first item in Sections 2 and 3. We now focus on mechanisms for the second item.

Runtime enforcement. In some cases, if some of the system components have been compromised, the remaining components can still be used for control with (potential) performance degradation but strong safety guarantees [35, 36]. On the other hand, when some components are compromised, it is necessary to rely on architectural support to ensure safe system operation in the case of attacks [33, 35] through a set of actuators. However, without architectural support, even when these actuators are identified they can continue to force the system into an unsafe state. This can be prevented with the use of secure/trusted hardware and architectural design that allows for decoupling of the attacked actuators. Similarly, if a compromised control module (e.g., a task running on an ECU) is detected, rebooting the controller and restoring it to a safe cyber-physical state could neutralize the attack; similarly, the system may decide to switch to a trusted controller that is safe, but may not be optimized for performance (e.g., as in the standard simplex architecture [32,39]). In situations where some, but not all of the control components are compromised, an interesting problem to investigate is the use of *micro-rebooting* for system recovery, which has shown to significantly reduce recovery cost, such as time to recover [10]. To design successful techniques, it is critical to have clear understanding of the underlying system architecture and how architectural support can be exploited to provide safe system performance even in the presence of attacks. Thus, one can clearly capture platform resources in the form of real-time, assume/guarantee properties of the sensor, controller and actuation modules. This will help support compositional analysis from the perspective of evolving software with runtime changes in the system configurations. In this context, a key aspect that will provide dynamic assurance is a clear formulation of various recovery and enforcement mechanisms at the level of individual agents as part of the agent's architecture. Another area to investigate is developing *repair transducers* for enforcing security policies. String transducers are automata that map input strings to output strings, and have been studied in the context of string *sanitization*.

Compositional Design. The monitoring techniques for secure and privacy-preserving CPS discussed in Section 3 will provide different resiliency guarantees for specific attack vectors, including claims about what attacks are detectable, identifiable, and can be attenuated through resilient control. A missing link is still providing guarantees on time-to-detection and identification, as well as potential control-performance cost degradation due to their use. One way to develop compositional design methods for combining different monitors is a hierarchical monitoring system, in order to improve system resiliency to attacks over any monitor individually. Intuitively, multiple deployed monitors can *guard* one another's blind spot or they can be activated at different time-instances due to the

constrained system (e.g., computation) resources. There are multiple challenges to tackle, e.g., modeling, types of assumptions and guarantees, implementation and performance degradation costs, as well as attackers' impact over time if the security-aware module is not active. The use of logic-based modeling and reasoning should also be investigated.

5 Conclusion

In this paper, we focused on the pressing need to bring about a paradigm shift in software development for multi-agent CPS. We sketched our position on three different orthogonal fronts to tackle the problem, namely, (1) designing specification languages that can capture both security and CPS aspects of systems, (2) runtime monitoring of CPS to detect security violations and detect attackers' attempts to compromise security and/or privacy, and (3) runtime enforcement to ensure security and safety of CPS. Our view is that security and privacy policies should be made a critical ingredient of agent interfaces with a goal of ensuring both localized safety and privacy for each agent, as well as guaranteeing global system safety and security. This is especially crucial and challenging in multi-agent CPS.

Acknowledgment

This research has been partially supported by the NSF SaTC-1813388, a grant from Iowa State University, NSF CNS-1652544 and the ONR under agreements number N00014-17-1-2012 and N00014-17-1-2504.

References

1. S. Agrawal and B. Bonakdarpour. Runtime verification of k-safety hyperproperties in hyperltl. In *Proceedings of the 20th IEEE Computer Security Foundations Symposium, CSF*, pages 239–252, 2016.
2. Y. S. R. Annapureddy, C. Liu, G. E. Fainekos, and S. Sankaranarayanan. S-TaLiRo: A tool for temporal logic falsification for hybrid systems. In *Tools and algorithms for the construction and analysis of systems*, volume 6605 of *LNCS*, pages 254–257. Springer, 2011.
3. E. Bartocci, J. V. Deshmukh, A. Donzé, G. E. Fainekos, O. Maler, D. Nickovic, and S. Sankaranarayanan. Specification-based monitoring of cyber-physical systems: A survey on theory, tools and applications. In *Lectures on Runtime Verification - Introductory and Advanced Topics*, pages 135–175. 2018.
4. S. Berkovich, B. Bonakdarpour, and S. Fischmeister. Runtime verification with minimal intrusion through parallelism. *Formal Methods in System Design*, 46(3):317–348, 2015.
5. M. Blaze, S. Kannan, I. Lee, O. Sokolsky, J. M. Smith, A. D. Keromytis, and W. Lee. Dynamic trust management. *Computer*, 42(2):44–52, 2009.
6. B. Bonakdarpour and B. Finkbeiner. Runtime verification for hyperLTL. In *Proceedings of the 16th International Conference on Runtime Verification (RV)*, pages 41–45, 2016.

7. B. Bonakdarpour and B. Finkbeiner. The complexity of monitoring hyperproperties. In *Proceedings of the 31st IEEE Computer Security Foundations Symposium (CSF)*, pages 162–174, 2018.
8. B. Bonakdarpour, C. Sanchez, and G. Schneider. Monitoring hyperproperties by combining static analysis and runtime verification. In *nternational Symposium On Leveraging Applications of Formal Methods, Verification and Validation (ISoLA)*, 2018. To appear.
9. N. Brett, U. Siddique, and B. Bonakdarpour. Rewriting-based runtime verification for alternation-free hyperltl. In *Proceedings of the 23rd International Conference on Tools and Algorithms for the Construction and Analysis of Systems (TACAS)*, pages 77–93, 2017.
10. G. Candea, S. Kawamoto, Y. Fujiki, G. Friedman, and A. Fox. Microreboot-a technique for cheap recovery. In *OSDI*, volume 4, pages 31–44, 2004.
11. S. Checkoway, D. McCoy, B. Kantor, D. Anderson, H. Shacham, S. Savage, K. Koscher, A. Czeskis, F. Roesner, T. Kohno, et al. Comprehensive experimental analyses of automotive attack surfaces. In *USENIX Security Symposium*. San Francisco, 2011.
12. M. R. Clarkson, B. Finkbeiner, M. Koleini, K. K. Micinski, M. N. Rabe, and C. Sánchez. Temporal Logics for Hyperproperties. In *Principles of Security and Trust (POST)*, pages 265–284, 2014.
13. J. Deshmukh, M. Horvat, X. Jin, R. Majumdar, and V. S. Prabhu. Testing cyber-physical systems through bayesian optimization. *ACM Trans. Embed. Comput. Syst.*, 16(5s):170:1–170:18, Sept. 2017.
14. J. V. Deshmukh, A. Donzé, S. Ghosh, X. Jin, G. Juniwal, and S. A. Seshia. Robust online monitoring of Signal Temporal Logic. *Formal Methods in System Design*, 2017.
15. J. V. Deshmukh, X. Jin, J. Kapinski, and O. Maler. Stochastic local search for falsification of hybrid systems. In *Automated Technology for Verification and Analysis - 13th International Symposium, ATVA 2015, Shanghai, China, October 12-15, 2015, Proceedings*, pages 500–517, 2015.
16. A. Dokhanchi, B. Hoxha, and G. Fainekos. On-line monitoring for temporal logic robustness. In *RV*, pages 231–246. 2014.
17. B. Finkbeiner, C. Hahn, M. Stenger, and L. Tentrup. Monitoring hyperproperties. In *Proceedings of the 17th International Conference on Runtime Verification (RV)*, pages 190–207, 2017.
18. A. Greenberg. Hackers remotely kill a jeep on the highway?with me in it. *Wired*, 7:21, 2015.
19. S. Jaksic, E. Bartocci, R. Grosu, R. Kloibhofer, T. Nguyen, and D. Nickovic. From signal temporal logic to FPGA monitors. In *13th ACM/IEEE International Conference on Formal Methods and Models for Codesign (MEMOCODE)*, pages 218–227, 2015.
20. X. Jin, J. V. Deshmukh, J. Kapinski, K. Ueda, and K. Butts. Powertrain Control Verification Benchmark. In *Proc. of Hybrid Systems: Computation and Control*, pages 253–262, 2014.
21. X. Jin, A. Donzé, J. V. Deshmukh, and S. A. Seshia. Mining Requirements from Closed-loop Control Models. In *Proc. of Hybrid Systems: Computation and Control*, 2013.
22. I. Jovanov and M. Pajic. Relaxing integrity requirements for resilient control systems. *CoRR*, abs/1707.02950, 2017.

23. I. Jovanov and M. Pajic. Sporadic data integrity for secure state estimation. In *2017 IEEE 56th Annual Conference on Decision and Control (CDC)*, pages 163–169, Dec 2017.
24. J. Kapinski, J. V. Deshmukh, X. Jin, H. Ito, and K. Butts. Simulation-based approaches for verification of embedded control systems: An overview of traditional and advanced modeling, testing, and verification techniques. *IEEE Control Systems*, 36(6):45–64, Dec 2016.
25. C. Kolias, G. Kambourakis, A. Stavrou, and J. Voas. Ddos in the iot: Mirai and other botnets. *Computer*, 50(7):80–84, 2017.
26. K. Koscher, A. Czeskis, F. Roesner, S. Patel, T. Kohno, S. Checkoway, D. McCoy, B. Kantor, D. Anderson, H. Shacham, et al. Experimental security analysis of a modern automobile. In *Security and Privacy (SP), 2010 IEEE Symposium on*, pages 447–462. IEEE, 2010.
27. V. Lesi, I. Jovanov, and M. Pajic. Network scheduling for secure cyber-physical systems. In *2017 IEEE Real-Time Systems Symposium (RTSS)*, pages 45–55, Dec 2017.
28. V. Lesi, I. Jovanov, and M. Pajic. Security-aware scheduling of embedded control tasks. *ACM Trans. Embed. Comput. Syst.*, 16(5s):188:1–188:21, Sept. 2017.
29. J. Li, P. Nuzzo, A. Sangiovanni-Vincentelli, Y. Xi, and D. Li. Stochastic contracts for cyber-physical system design under probabilistic requirements. In *ACM/IEEE Int. Conf. on Formal Methods and Models for System Design*, 2017.
30. G. Liang, S. R. Weller, J. Zhao, F. Luo, and Z. Y. Dong. The 2015 ukraine blackout: Implications for false data injection attacks. *IEEE Transactions on Power Systems*, 32(4):3317–3318, 2017.
31. O. Maler and D. Nickovic. Monitoring temporal properties of continuous signals. In *Proceedings of FORMATS-FTRTFT*, volume 3253 of *LNCS*, pages 152–166, 2004.
32. 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 *Proceedings of the 2nd ACM international conference on High confidence networked systems*, pages 65–74. ACM, 2013.
33. M. Pajic, I. Lee, and G. J. Pappas. Attack-resilient state estimation for noisy dynamical systems. *IEEE Transactions on Control of Network Systems*, 4(1):82–92, March 2017.
34. M. Pajic, R. Mangharam, G. J. Pappas, and S. Sundaram. Topological conditions for in-network stabilization of dynamical systems. *IEEE Journal on Selected Areas in Communications*, 31(4):794–807, April 2013.
35. 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 Systems*, 37(2):66–81, April 2017.
36. M. Pajic, J. Weimer, N. Bezzo, P. Tabuada, O. Sokolsky, I. Lee, and G. Pappas. Robustness of attack-resilient state estimators. In *ACM/IEEE International Conference on Cyber-Physical Systems (ICCPS)*, pages 163–174, April 2014.
37. S. Savage. Modern automotive vulnerabilities: Causes, disclosures, and outcomes. 2016.
38. J. Schumann, P. Moosbrugger, and K. Y. Rozier. R2u2: monitoring and diagnosis of security threats for unmanned aerial systems. In *Runtime Verification*, pages 233–249. Springer, 2015.
39. D. Seto, B. H. Krogh, L. Sha, and A. Chutinan. Dynamic control system upgrade using the simplex architecture. *IEEE Control Systems*, 18(4):72–80, 1998.

40. S. Sundaram, M. Pajic, C. Hadjicostis, R. Mangharam, and G. Pappas. The Wireless Control Network: Monitoring for Malicious Behavior. In *49th IEEE Conference on Decision and Control (CDC)*, pages 5979–5984, Dec 2010.
41. S. Sundaram, S. Revzen, and G. Pappas. A control-theoretic approach to disseminating values and overcoming malicious links in wireless networks. *Automatica*, 48(11):2894–2901, 2012.
42. A. G. West, A. J. Aviv, J. Chang, V. S. Prabhu, M. Blaze, S. Kannan, I. Lee, J. M. Smith, and O. Sokolsky. Quantm: A quantitative trust management system. In *Proceedings of the Second European Workshop on System Security*, pages 28–35. ACM, 2009.