Safety, Security, and Reliability: The Automotive Robustness Problem and an Architectural Solution

Sandip Ray
Department of Electrical and Computer Engineering
University of Florida at Gainesville
Gainesville, FL 32611. USA
sandip@ece.ufl.edu

***Abstract*—As we move towards increasingly autonomous vehicles, it is getting increasingly crucial to ensure that they behave safely, securely, and reliably. Automotive *robustness* refers to the study of synergies and trade-offs between these requirements. In this paper we discuss challenges and practice in two critical components of automotive robustness, functional safety and security. We also discuss the potential application of a flexible architecture for systematically implementing these requirements.**

I. Introduction

Automotive systems are increasingly turning into complex, distributed cyber-physical systems. A modern automotive has over 100 computing elements (often referred to as "Electronic Control Unit" or ECU), about 100MB of software, and several in-vehicle networks, in addition to a diversity of sensors and actuators. Electronics and software today account for more than 50% of the design overhead of a vehicle and represent the key factors to provide market distinction and difference in price points [1], [2]. The trend is towards a sharper increase in electronic and computing components as we move towards vehicles with greater and greater autonomy. Indeed, a fast proliferation of autonomous vehicles is often welcomed as it comes with a promise of safer transportation. More than 90% of accidents on road today happen because of human errors, resulting, in the United States alone, in over 50 million serious injuries and a cost of over 3 trillion dollars every year [3]. Reducing and removing the human factor from driving consequently promises to provide a transformative change in road safety.

On the other hand, autonomy can meet the promise of safety only if the electronics and software used to replace human actions are themselves safe, secure, and reliable. Unfortunately, this has not been the case traditionally. Software and hardware have often been riddled with bugs, vulnerabilities, and malfunctions, which have been discovered or exploited in-field sometimes resulting in catastrophic consequences. The situation is exacerbated with the very trend of autonomy, which results in a corresponding sharp increase in complexity of electronic and computing components [4], [5]. For instance, an autonomous car will require several components not required in today's vehicles, including real-time vehicle-to-vehicle (V2V) and vehicle-to-infrastructure (V2I) communications with a variety of networks of different levels of trustworthiness, a diversity of sensors to detect driving conditions (*e.g.*, potholes, moisture, pedestrians, etc.), and distributed computing elements to perform in-vehicle analytics and mitigation technology to react to evolving conditions on the fly [6]. Consequently, it is becoming increasingly non-trivial, in some cases infeasible, to explore, analyze, and validate the diverse effects of electronic components, and an "innocent" misconfiguration or error in one component can result in a catastrophic and complete failure of the entire system. The following "tongue-in-cheek" quote, attributed to Paar, succinctly summarizes the situation [1]:

> If vehicles were developed in the same manner as telecommunications, then an average car would reach top speeds of 10^9km/h at 400M HP and the car would be hacked four times a year.

The quotation is from a time when automotive systems were still considered mechanical or at best electro-mechanical rather than electronic systems. However, today, vehicles *are* being developed as electronic systems! Consequently, one must account for the corresponding different trajectory of automation in a modern automobile, and additionally reconcile this growth and its accompanying system vulnerabilities) with the expectations of trustworthiness and robustness that customers have traditionally expected out of a car.

Automotive *robustness* is the subject that studies safety, security, and reliability of automotive systems. It is one of the critical research topics as we move towards connected cars, and is one of the crucial stumbling blocks in our quest towards the adoption of autonomous cars.

The goal of this paper is to provide a general introduction to the area of automotive robustness. We discuss the scope and spectrum of automotive robustness, introduce the various *robustness policies* that must be enforced, and the current state of the practice (and its limitations). We also discuss some emergent research towards developing a robust architecture for future automotive systems.

The remainder of the paper is organized as follows. Section II introduces the various robustness components, and Section III introduces robustness policies. Section IV discusses the state of the practice in enforcing automotive robustness. In Section V, we discuss an emergent architecture that we are developing for implement robustness requirements. We discuss related work in Section VI and conclude in Section VII.

II. The Automotive Robustness Problem

Robustness requirements for modern automotive systems can be roughly divided into the following three categories, which together form the robustness triad.

1) **Safety.** Roughly, safety requirements specify that a car should not harm any other agent in its environment due to hazards caused by malfunctioning of electrical/electronic (E/E) components.
2) **Security.** Security refers to the requirement that the electronic components of the car must be resilient against system hacks.
3) **Reliability.** This refers to robustness of electro-mechanical and mechanical components, and their interfaces with the E/E components of the system.

Obviously, each component of the triad entails several complex requirements. For automotive systems, functional safety requirements are embodied by the ISO 26262 standard [7]. It defines five risk levels called Automotive Safety Integrity Levels (ASIL) to identify and grade severity of various hazardous events. Functional safety also requires the system to operate correctly under certain systematic and transient hardware faults. Security constraints identify various authentication requirements for communications received by the automotive from a variety of networks with different levels of trustworthiness. These constraints must account for trade-offs between security and privacy (*e.g.*, a strong authentication requirement may compromise the privacy of the sender), between security and real-time requirements (*e.g.*, high computational complexity necessary for strong authentication may conflict with hard real-time requirements), etc. Security constraints also account for access control and propagation restrictions on various system assets (*e.g.*, cryptographic keys, proprietary firmware, debug modes, etc.) at various points in the system life-cycle. Finally, reliability requirements include early warning against component wear-outs, mechanisms to ensure slow and gradual degradation, etc. [6]

III. Robustness Policies

Mapping the high-level robustness constraints to various design and implementation requirements is obviously a non-trivial exercise. In practice, this is achieved by defining a large number of robustness policies. The following are representative policy examples for current and emergent automotive systems.

- **Example 1:** If detected moisture in roadway exceeds a threshold, then turn off ECO mode and enable safe braking.
- **Example 2:** Messages received through V2V module must be authenticated by the authentication module before any further processing.
- **Example 3:** A V2V communication from highway patrol must be authenticated and responded to within a (specified) upper bound of time.

Example 1 is a functional safety constraint and Examples 2 and 3 are security constraints. Robustness policies are defined as actionable items that can be processed by architects and designers to create a final implementation. On the other hand, robustness requirements can be complicated in subtle ways. Example 3 is an interesting example of the interplay between security, reliability, and real-time needs. In particular, the upper bound on response time for a communication from highway patrol implicitly imposes an upper bound on the computation and communication of the authentication module. Other robustness policies can involve protection against component aging, handling security issues caused by complex supply-chain management, etc.

IV. Robustness Policy Enforcement in Practice

Today's industrial practice depends primarily on human insight for policy definition, analysis, and implementation. To facilitate that, robustness policies are typically embodied into a document called Safety, Security, Reliability document (SSR for short). In an SSR, the robustness constraints are specified by a collection of graphs, charts, tables, and message flows, coupled with English text [8]. The SSR is used by architects and designers as a guide to architect, design, implement, and validate robustness constraints in the same manner as a design or architecture document is used to implement functionality. Since the documents are not formal, these constraints cannot be subjected to any mechanical analysis before implementation. On the other hand, it leaves open the possibility of inconsistencies in the requirements themselves. For example, the security vs. real-time trade-off discussed above may not be consistently resolved in the presence of other constraints that require strong (and hence computationally intensive) message authentication. Such problems are typically discovered late in the development and can be extremely expensive since fixing them may require involvement of several players in the complex automotive supply-chain. Even in the absence of such inconsistencies, it is difficult to ensure that the implementation of the policies indeed conforms to robustness requirements. Note that each robustness policy may implicitly involve a number of ECUs, sensors, actuators, and communication between them. For instance, the safety policy in Example 1 is a policy that uses the sensory information from the wheel and enforces an actuarial condition on the brake. ECUs connected to the relevant sensors and actuators must coordinate to enforce this policy. Consequently, validation of the policy will also require comprehension of the operations of different ECUs, how they process sensory and actuarial information, and how they communicate with each other. Doing this for policies involving multiple ECUs and sensors is difficult. Correspondingly, since they are implemented in an ad hoc manner, upgrading a robustness policy in field (potentially in response to changing requirements during the long life-time of the system) is also infeasible [9].

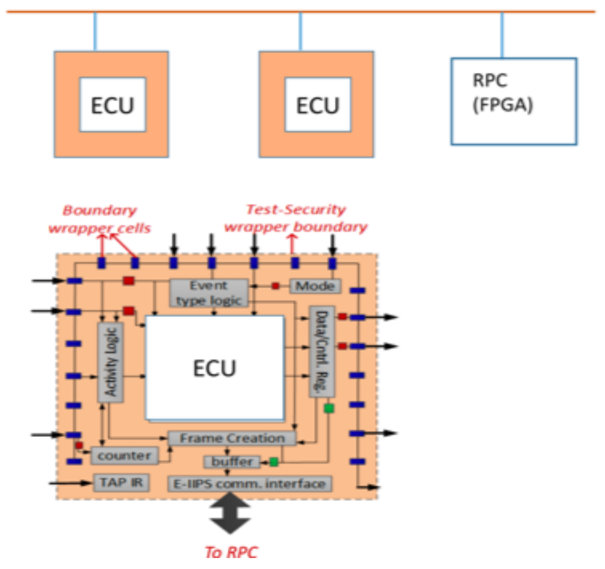

Fig. 1. A Centralized Policy Implementation Architecture. **(Top)** RPC is a standalone unit connected to the in-vehicle network. **(Bottom)** The ECUs are augmented with smart wrappers that augment the traditional test wrappers.

V. An Emergent Framework for Automotive Robustness

We are developing an architectural framework for systematic implementation of diverse robustness requirements. The key idea of the framework is to develop plug-and-play hardware modules that implement and controls all the robustness requirements. In this section, we provide a brief overview of this architecture. However, the reader should consider the comments in this section with the caveat that the architecture is under active development and any discussion on viability is merely speculative.

Fig. 1 shows the basic approach for our proposed architecture. The idea is to have a standalone, plug-and-play reconfigurable hardware unit dedicated to implementing robustness policies. The hardware unit, called Robustness Policy Controller ("RPC" for short) is connected directly to an in-vehicle network, such as CAN, like other ECUs. However, instead of having any sensors or actuators attached to it, is only responsible for robustness policy enforcement. Recall from Section III that a robustness policy may involve coordination of several ECUs with diverse sensory and actuarial information. To enable enforcement of such requirement, the RPC is notified of events that pertain to the implemented policies, *e.g.*, for Example 1 in Section III, RPC is notified when the ECU controlling the wheel obtains sensory information indicating high moisture level. This can be achieved by augmenting the ECUs with a standardized wrapper implementation, extending the already existing test wrappers. The goal of the wrapper is to detect events pertaining to any of the implemented policies and notify RPC of the occurrence of that event. The wrappers are designed to be "smart", *i.e.*, they should only communicate relevant events while ensuring that all policies are enforced. Finally, they are configurable, *i.e.*, if additional events are necessary then it should be possible to capture them without overhauling the entire implementation.

How can we implement smart wrappers with all the above requirements? To facilitate use of the wrappers in different execution modes, RPC will configure the wrappers at boot time with relevant events. Furthermore, we can interface the wrapper with the debug interface of the ECU to enable observability and controllability of events not previously anticipated, perhaps because of policy update after deployment. All ECUs in practice include a post-silicon debug interface to enable debug and repair of bugs found in field. These interfaces therefore provide observability and controllability of a significant number of internal events. In recent experiments, we found that an interface to debug together with a broad class of generic events makes it possible to extract most relevant events for enforcing a spectrum of security policies. We suspect that this insight carries over to arbitrary robustness policies as well, in particular since these events are often controlled through over-the-air updates.

The final piece of the puzzle is the RPC implementation itself. It must be a plug-and-play hardware module that can implement arbitrary policies. Furthermore, it must allow reconfigurability to enable in-field updates. Our solution is to use an embedded field-programmable gate array (FPGA). FPGAs have recently found significant application in servers and data centers on the one hand, and Internet-of-Things on the other, because of their amenability to in-field reconfiguration. Note that since policy control essentially entails a guarded state machine, it can be easily implemented in FPGA. Furthermore, for security and in-field update, bit-streams can be encrypted. We are also investigating FPGA virtualization paradigms that provide further protection for FPGA implementations while enabling smooth update capability.

We end the discussion on the centralized architecture framework with a brief note on verification. As we remarked before, policy implementations are hard to verify since they involve multiple ECUs. However, given the centralized approach, it is sufficient only to verify the RPC module and its interfaces with the security wrappers, making such verification feasible for practical systems [10].

VI. Related Work

There has been significant recent work on safety, security, and reliability of various CPS applications [11], [12], [13]. Most of these approaches consider specific robustness requirements (*e.g.*, safety, reliability, security), and provide mitigation techniques for these requirements. In particular, we have not seen any research on formalization, abstraction, or mitigation techniques that target arbitrary robustness policies. For automotive systems in particular, significant attention has been paid recently to security properties. However, the focus has been on identifying and demonstrating security vulnerabilities rather than on developing systematic architecture for their mitigation. Recently, there has been attempts at developing comprehensive automotive architecture. However, these architectures primarily focus systematic V2V communication and hardware/software co-design [14], [15].

Our work on automotive robustness architecture has been inspired by our previous work on developing architecture for systematic implementation of System-on-Chip (SoC) security policies [16], [17], [18], [10], [19]. This work showed how to systematically implement SoC security policies using a plug-and-play hardware module. However, the policies investigated were purely digital with no CPS component. Furthermore, the approach did not consider policies involving multiple control units connected through a network.

VII. Conclusion and Future Work

We have provided an overview of automotive robustness problem, outlined the scope of the problem and challenges, and discussed the current state of the practice and its limitations. In spite of its critical importance, systematizing design and implementation of robustness requirements has not received significant attention from the research community. We hope that the presentation of this paper will help spur discussions, exploration, and research in this area.

We have presented an emergent direction for architecting robustness policies, but our approach has only scratched the surface. One critical direction for future work is developing a distributed implementation of the controller architecture. Note that while the centralized architecture is conceptually simple, it may suffer from challenges with congestion since events need to be communicated to RPC for policy enforcement. This is particularly significant in the presence of real-time constraints when the constraints may be violated by slow data movement because of congestion in the in-vehicle networks. The goal of the distributed implementation is to minimize communication and provide a more efficient policy enforcement through collective, local enforcement at individual ECUs. In addition, we are also considering approaches to automated policy extraction from SSRs, and an automated CAD flow for synthesizing policies to an RPC design.

References

[1] A. Weimerkirsch, "Automotive and Industrial Data Security," in *Cybersecurity and Cyber-physical Systems Workshop*, 2012.

[2] S. Ray, W. Chen, J. Bhadra, and M. A. A. Faruque, "Extensibility in Automotive Security: Current Practice and Challenges," in *DAC 2017*, 2017.

[3] National Highway Traffice Safety Association, "National Motor Vehicle Crash Causation Survey," See URL: https://crashstats.nhtsa.dot.gov/Api/Public/ViewPublication/811059.

[4] National Highway Traffic Safety Association, "Motor Vehicles Recall," See URL: https://www.recalls.gov/nhtsa.html.

[5] C. Miller and C. Valasek, "Remote Exploitation of an Unaltered Passenger Vehicle," in *BlackHat USA*, 2015.

[6] D. Pradhan, "Solving the Automotive Dilemma," in *Microprocessor Test & Verification (MTV 2016)*, 2016.

[7] International Standardization Organization, "ISO 26262-1:2011 Road Vehicles Functional Safety," 2011.

[8] L. Regers, "The Road Ahead for Securely Connected Cars," in *IEEE International Solid State Circuits Conference*, 2016.

[9] S. Ray, A. Basak, and S. Bhunia, "Patching the Internet of Things," *IEEE Spectrum*, vol. 54, no. 11, pp. 31–35, 2017.

[10] A. P. D. Nath, S. Bhunia, and S. Ray, "ArtiFact: Architecture and CAD Flow for Efficient Formal Verification of SoC Security Policies," in *ISVLSI 2018*, 2018.

[11] M. Rungger and P. Tabuada, "A Symbolic Approach to the Design of Robust Cyber-physical Systems," in *52nd IEEE Conference on Decision and Control*, 2013.

[12] Q. Zhu and T. Basar, "Robust and Resilient Control Design for Cyber-Physical Systems with an Application to Power Systems," in *IEEE Conference on Decision and Control and European Control Conference*, 2011.

[13] F. Hu, Y. Lu, A. Vasilakos, Q. Hao, R. Ma, Y. Patil, T. Zhang, J. Lu, X. Li, , and N. Xiong, "Robust Cyber-physical Systems: Concepts, Models, and Implementation," *Future Generation Computer Systems*, vol. 56, pp. 449–475, 2016.

[14] NXP Semiconductors, "The 4+1 Layer Security Architecture," http://www.nxp.com/assets/documents/data/en/white-papers/MULTI-LAYER-VEHICLE-SECURITY-WP.pdf.

[15] F. Sagstetter, M. Lukasiewycz, S. Steinhorst, M. Wolf, A. Bouard, S. Jha, and S. Chakraborty, "Security Challenges in Automotive Hardware/software Architecture Design," in *Design Automation and Test in Europe*, 2013.

[16] A. Basak, S. Bhunia, and S. Ray, "A Flexible Architecture for Systematic Implementation of SoC Security Policies," in *ICCAD*, 2015.

[17] ——, "Exploiting design-for-debug for flexible SoC security architecture," in *DAC*, 2016.

[18] S. Ray, J. Yang, A. Basak, and S. Bhunia, "Correctness and Security at Odds: Post-silicon Validation of Modern SoC Designs," in *DAC*, 2015.

[19] A. P. D. Nath, S. Ray, A. Basak, and S. Bhunia, "An Architecture and CAD Flow for Hardware Patch," in *ASPDAC*, 2017.